НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.04

«До захисту допущено»
Завідувач кафедри
_____Едуард ЖАРІКОВ
«___»_____2022 р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра
за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»
зі спеціальності 121 «Інженерія програмного забезпечення»
на тему: «Програмне забезпечення для системи обробки електронних документів»

Виконав:
студент ІІ курсу, групи ІП-13мп
Шкурко Денис Олександрович
_____
Керівник:
Доцент, к.т.н., доц.,
Фіногенов Олексій Дмитрович
_____
Рецензент:
доцент кафедри ІСТ, к.т.н., доц.,
Лісовиченко Олег Іванович
_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Шкурко Денис Олекандрович

Київ – 2022 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Едуард ЖАРІКОВ

«___»_____2022р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Шкурку Денису Олександровичу

1. Тема дисертації «Програмне забезпечення для системи обробки електронних документів», науковий керівник дисертації Фіногенов Олексій Дмитрович, к.т.н., доц., затверджені наказом по університету від «09» листопада 2022 р. № 4115-с

2. Термін подання студентом дисертації «09» грудня 2022 р.

3. Об’єкт дослідження – універсальне сканування та шаблонізатор, а також відмінювання антропонімів.

4. Предмет дослідження – є методи сканування та створення шаблонів, а також особливості відмінювання антропонімів, як українського, так і іноземного походження.

5. Перелік завдань, які потрібно розробити: знайти та проаналізувати існуючі рішення, обрати архітектуру сканування та методи створення шаблонів, дослідити та створити алгоритм відмінювання антропонімів, реалізувати програмне забезпечення

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – 4 плакати

7. Орієнтовний перелік публікацій – одна публікація

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Перевірка на співпадіння	доцент Лісовиченко О.І.		

9. Дата видачі завдання «01» вересня 2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Переддипломна практика	01.09.2022 - 24.10.2022	
2	Виконання експериментальних досліджень	25.09.2022 - 31.10.2022	
3	Розробка стартап-проекту	01.11.2021 - 10.11.2021	
4	Підготовка доповіді на конференції	11.11.2021 – 20.11.2021	
5	Оформлення пояснювальної записки	21.11.2022 - 05.12.2022	
6	Подання дисертації на попередній захист	22.11.2022	
7	Подання дисертації на захист	09.12.2022	

Студент Денис ШКУРКО

Науковий керівник Олексій ФІНОГЕНОВ

РЕФЕРАТ

Розмір пояснювальної записки – 61 аркушів, містить 20 ілюстрацій, 26 таблиці, 3 додатки, 10 посилань на джерела.

Актуальність теми. У роботі розглянуто створення додаткового програмного забезпечення, що дозволить пришвидшити або автоматизувати роботу систем, пов’язаних із оборотом електронних документів.

Мета дослідження. Написання зручного програмного забезпечення для сканування та створення шаблонів, що дозволить з легкістю налаштовувати ці продукти для будь-якої системи документообігу. Також окремою ланкою виступає дослідження відмінювання антропонімів та реалізація скрипту для виконання цієї задачі.

Об’єкт дослідження: універсальне сканування та шаблонізатор, а також відмінювання антропонімів.

Предмет дослідження: методи сканування та створення шаблонів, а також особливості відмінювання антропонімів, як українського, так і іноземного походження.

Для реалізації поставленої мети **сформульовані наступні завдання**:

- знайти та проаналізувати існуючі рішення;
- обрати архітектуру сканування та методи створення шаблонів;
- дослідити та створити алгоритм відмінювання антропонімів;
- реалізувати програмне забезпечення.

Наукова новизна полягає у розробці універсального програмного забезпечення, що досить легко налаштувати під систему обробки електронних документів, для сканування та створення шаблонів. Побудований алгоритм для відмінювання антропонімів не має аналогів у швидкості та простоті і є незалежним від обчислювальної швидкості.

Практичне значення отриманих результатів полягає в тому, що розроблене програмне забезпечення може бути використане для

пришвидшення або автоматизування роботи систем, пов’язаних із оборотом електронних документів.

Зв’язок з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського".

Апробація. Наукові положення дисертації пройшли апробацію на Третій Всеукраїнській науково-практичній конференції молодих вчених та студентів «ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ І ПЕРЕДОВІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ» (SoftTech-2022 осінь) – м. Київ.

Публікації. Наукові положення дисертації опубліковані в:

1. Шкурко Д.О. АВТОМАТИЧНЕ ВІДМІНЮВАННЯ АНТРОПОНІМІВ В ОФІЦІЙНИХ ДОКУМЕНТАХ / Д.О. Шкурко, О.Д. Фіногенов // Матеріали Третьої Всеукраїнської науково-практичної конференції молодих вчених та студентів «ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ І ПЕРЕДОВІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ» (SoftTech-2022 осінь) – м. Київ: НТУУ «КПІ ім. Ігоря Сікорського», 23-25 листопада 2022 р.

Ключові слова: ДОКУМЕНТООБІГ, ШАБЛОНІЗАТОР, СКАНЕР

ABSTRACT

Explanatory note size – 61 pages, contains 20 illustrations, 26 tables, 3 applications, 10 references.

Topicality. The work considers the creation of additional software that will allow speeding up or automating the operation of systems related to the circulation of electronic documents.

The aim of the study. Writing user-friendly scanning and template software that will allow you to easily customize these products for any document management system. Also, a separate link is the study of declension of anthropons and the implementation of a script to perform this task.

Object of research: universal scanning and template generator, as well as declension of anthroponyms.

The subject of the study: methods of scanning and creating templates, as well as features of declension of anthroponyms, both of Ukrainian and foreign origin.

To achieve this goal, the **following tasks** were formulated:

- find and analyze existing solutions;
- choose scanning architecture and pattern creation methods;
- investigate and create an anthropone declension algorithm;
- implement software.

The scientific novelty is that there are no universal programs that are quite easy to adjust to the system for scanning and creating templates. The constructed algorithm for declension of anthroponyms has no analogues in speed and simplicity and is independent of computing speed.

The practical value of the obtained results is that the developed software can be used to speed up or automate the work of systems related to the circulation of electronic documents.

Relationship with working with scientific programs, plans, topics. Work was performed at the Department of Informatics and Software Engineering of the

National Technical University of Ukraine «Kyiv Polytechnic Institute. Igor Sikorsky».

Approbation. The scientific provisions of the dissertation were tested at the Third All-Ukrainian scientific and practical conference of young scientists and students "SOFTWARE ENGINEERING AND ADVANCED INFORMATION TECHNOLOGIES" (SoftTech-2022 autumn) - Kyiv.

Publications. The scientific provisions of the dissertation were published in:

1) SHKURKO D.O. AUTOMATIC CHANGE OF ANTHROPONYMS IN OFFICIAL DOCUMENTS / D.O.Shkurko, O.D. Finohenov // Materials of the Third All-Ukrainian scientific and practical conference of young scientists and students "SOFTWARE ENGINEERING AND ADVANCED INFORMATION TECHNOLOGIES" (SoftTech-2022 autumn) - Kyiv: National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", November 23-25, 2022.

Keywords: Document menagment, template maker, scanner

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

XML - Extensible Markup Language

DOCX - Microsoft Word Open XML Document

ПІБ – прізвище, ім’я, по-батькові

JS – javaScript

БД – база даних

BPE – business process for enterprice

FTP – file transfer protocol

MQTT – message queue telemetry transport

ETS – erlang term storage

DETS – data erlang term storage

HTTP – hypertext transfer protocol

RPC – remote procedure call

ISO – international organization of standardization

ВСТУП

На сьогоднішній день зменшується кількість паперових носіїв інформації. Все більше документів зберігаються в електронному варіанті на накопичувачах. Тому виникає потреба в створенні додаткового програмного забезпечення або сервісів, що дозволять пришвидшити або автоматизувати роботу систем, пов’язаних із оборотом електронних документів. Це вплине також на якість роботи. Тому пропонується дослідити у магістерській дисертації питання автоматичного створення шаблонів. Проаналізувати відмінювання антропонімів, за сучасними правилами української мови, що забезпечить правову та нормативну силу юридичних документів. Окремим питанням постає задача сканування документів як локальними так і мережевими переферійними пристроями, це дозволить з легкістю отримувати зображення зі сканера без додаткового налаштування, окрім встановлення основного драйверу.

Метою дослідження є написання зручного програмного забезпечення для сканування та створення шаблонів, що дозволить з легкістю налаштовувати ці продукти для будь-якої системи документообігу. Також окремою ланкою виступає дослідження відмінювання антропонімів та реалізація скрипту для виконання цієї задачі.

Об’єктом дослідження є саме універсальне сканування та шаблонізатор, а також відмінювання антропонімів.

Предметом дослідження є методи сканування та створення шаблонів, а також особливості відмінювання антропонімів, як українського, так і іноземного походження.

Задачі, що повинні бути виконанні:

- знайти та проаналізувати існуючі рішення;
- обрати архітектуру сканування та методи створення шаблонів;
- дослідити та створити алгоритм відмінювання антропонімів;
- реалізувати програмне забезпечення.

Наукова новизна полягає у тому, що не існує універсальних програм, що досить легко налаштувати під систему для сканування та створення шаблонів. Побудований алгоритм для відмінювання антропонімів не має аналогів у швидкості та простоті і є незалежним від обчислювальної швидкості.

1 ВИКОРИСТАННЯ ТА ФУНКЦІОНУВАННЯ СИСТЕМИ ОБРОБКИ ЕДЕКТРОННИХ ДОКУМЕНТІВ

Для вирішення задачі створення пакету програмного забезпечення для полегшення електронної обробки документів, що в подальшому можуть бути використанні в системі електронного документообігу, потрібно розглянути наступні речі, а саме:

- шаблонізатор;
- програма для відмінювання антропонімів;
- сканер.

Шаблонізатор дозволить створювати шаблони у файлі docx формату та за допомогою можливостей бібліотеки OpenXML підставляти значення у документ із форми, це дозволяє скоротити час редагування документу, або ж взагалі автоматизувати цей процес, до є дуже важливим при обороті документів на базі підприємства.

Відмінювання антропонімів досить важлива задача у документах, часто виникає, що документ перед підписанням потрапляє на розробку, через синтаксичні помилки у ПІБ. Тому автоматичне відмінювання може скоротити кількість стадій проходження документу від розробки до підписання.

За допомогою сканера можна сканувати документ у систему, основною проблемою сканерів як мережевих, так і стаціонарних є їх налаштування та підключення до системи. Проте можна зробити шаблон, що скоротить час налаштування сканування.

1.1 Аналіз алгоритмів для підстановки значень у документи

З розвитком цифрової трансформації в сучасному світі, виникає потреба в автоматизації процесу документообігу. Це дозволить прискорити та полегшити роботу в даній сфері. Тому особливе значення починають

набувати шаблони вже готових документів, де вказаний текст підставляться із форми, яку вказує користувач системи, або з самої бази системи.

З існуючих програм досить мало виконують поставлені задачі підстановки значень у документи. Зазвичай алгоритм виглядає так, що ми у тексті docx ставимо теги по типу ${some information}(рис. 1.1), далі ж йде автозаміна по пошуку назви змінної з розпарсеної XML.

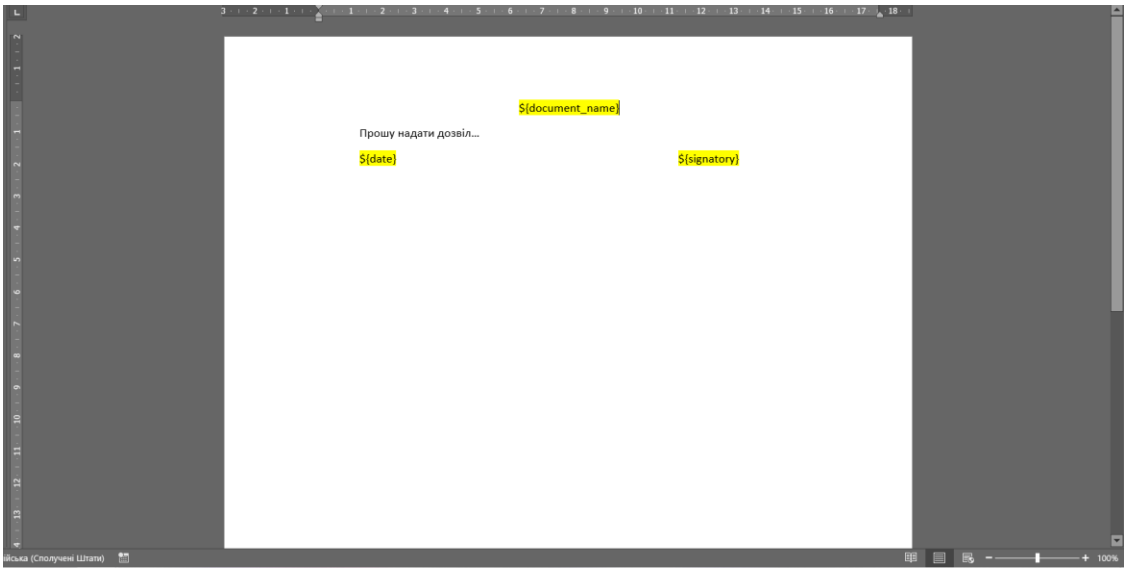


Рисунок 1.1 - Приклад тегу

Це дає змогу підставити будь які значення замість ключа(змінної), що дозволяє автоматизувати створення документів, або звітності, тощо.

Описаний алгоритм виконує свою функцію, але все досить неоднозначно. В даному випадку просто підставляється заміна значення, це дозволяє підставляти тільки текст, що досить складно для наповнення таблиць, і змінні залишаються завжди статичними. Для вирішення цього питання треба маніпулювати з XML, а саме замість тегу підставляти частину XML, що досить проблематично з обсягу універсальності шаблонізатору.

Тому критично буде зробити автоматизацію двигуна створення шаблону, де легко створюватимуться змінні у документі, при цьому тег легко змінювані та є можливість використувати як стилі так формат

підставлення(рис. 1.2). Також це дає змогу підставляти у документ як списки, так і таблиці у форматі csv, можливість підставляти зображення та, навіть, змінювати контитули.

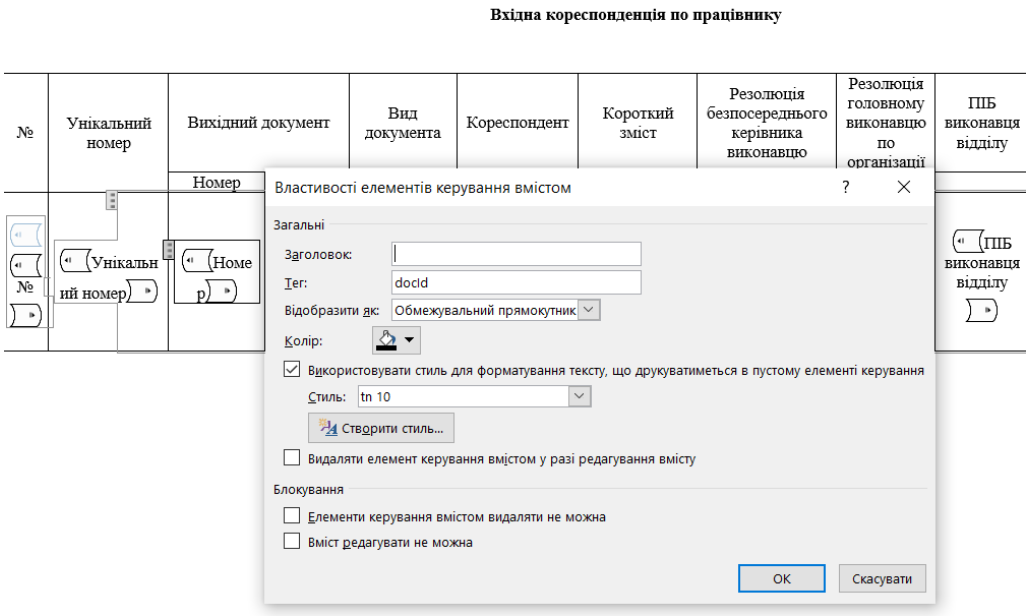


Рисунок 1.2 - Вигляд змінної у Microsoft office word

Це дозволить зменшити обсяг витраченого часу у створенні та реєстрації вихідних або внутрішній документів в системі. Дозволить швидко створювати звіти з даних, наданих запитом з бази даних.

1.2 Аналіз алгоритмів та програм для відмінювання антропонімів

Антропоніміка – основний розділ ономастики, її об'єкт вивчення – це антропоніми та власні особові назви. В свою чергу відмінювання антропонімів – це відмінювання прізвищ та імен громадян або учасників процесу документообігу. Досить важливо правильно відмінювати ПІБ, так як це є критичним моментом у достовірності, нормативної та юридичної сили документу. Так як в теперішній час обсяг електронних документів збільшується, постає потреба у автоматизації створення документів за

шаблонами, де у певні теги записуються прізвища, особові імена та патроніми. І за нормативними документами юридичних компаній є встановлені правила ведення документації та її обліку. І незначна помилка у антропонімі учасника документу може призвести до порушення цих правил. Тому виникає потреба у програмі, яка буде автоматично, швидко, а саме головне правильно, виконувати відмінювання власних особових назв.

Найпоширенішим вирішенням для даної проблеми є зберігання кожного відмінку антропоніма у базі даних, де саме ці данні вводить адміністратор системи.

З існуючих програм досить багато неправильно відмінюють винятки, або прізвища іноземного походження. Також є варіанти де ПІБ занесені до кластерів або звичайної БД, що досить ресурсозатратно з точки зору як часу пошуку, так і з точки зору матеріального забезпечення. Також повинні вводитися вручну данні невідомих ПІБ для подальшого відмінювання. Це робить зайве адміністрування системи.

Якщо взяти відмінювання, то сенс полягає у тому, що воно відбуває

Тому кращим варіантом буде описати за допомогою pattern matching всі випадки співставлення антропоніма з правилом української мови.

1.3 Аналіз алгоритмів для сканування документи

Сканування також є важливим елементом в системах обробки електронних документів. Швидкість і зручність програми сканування дозволяє легко маніпулювати зі сканером, буть то мережевий чи локальний. Однак підтримка відображення та обробка растових даних вимагає великих витрат. Адже потрібний як користувацький інтерфейс так вбудоване керування пристроєм, що надає широкий асортимент налаштування сканування доступних зображень.

За допомогою інтерфейсу відбувається взаємодія, а саме передача закодованого зображення, між сканером та принтером(рис. 1.3)

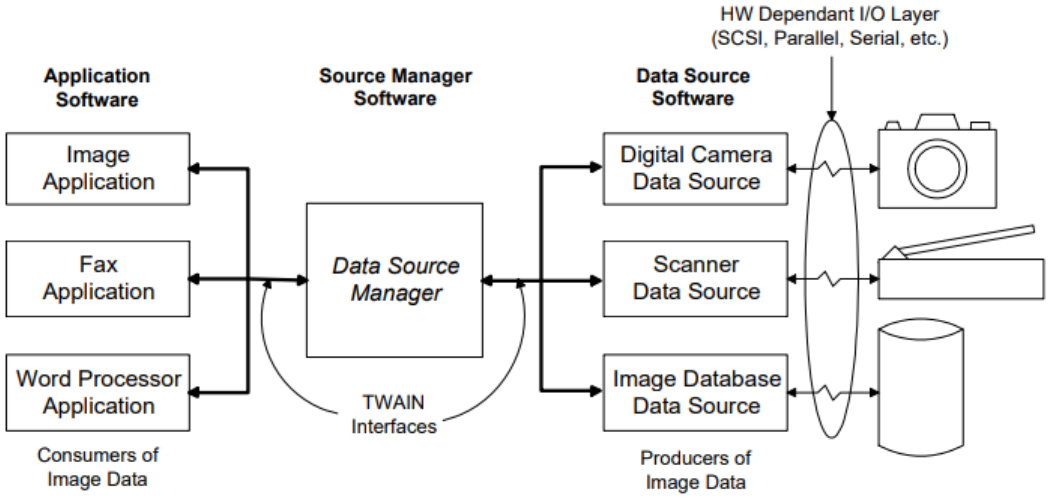


Рисунок 1.3 - Елементи взаємодії application та data sourse

Висновки до розділу

У першому розділі було розглянуто основні алгоритми, що вирішують задачі створення пакету програмного забезпечення для полегшення електронної обробки документів. Проаналізовано вже існуючі алгоритми та виявленні їх недоліки. Для покращення роботи системи виникає потреба у адаптивності та конкурентності додаткового програмного забезпечення, що більшість алгоритмів не виконує.

2 ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

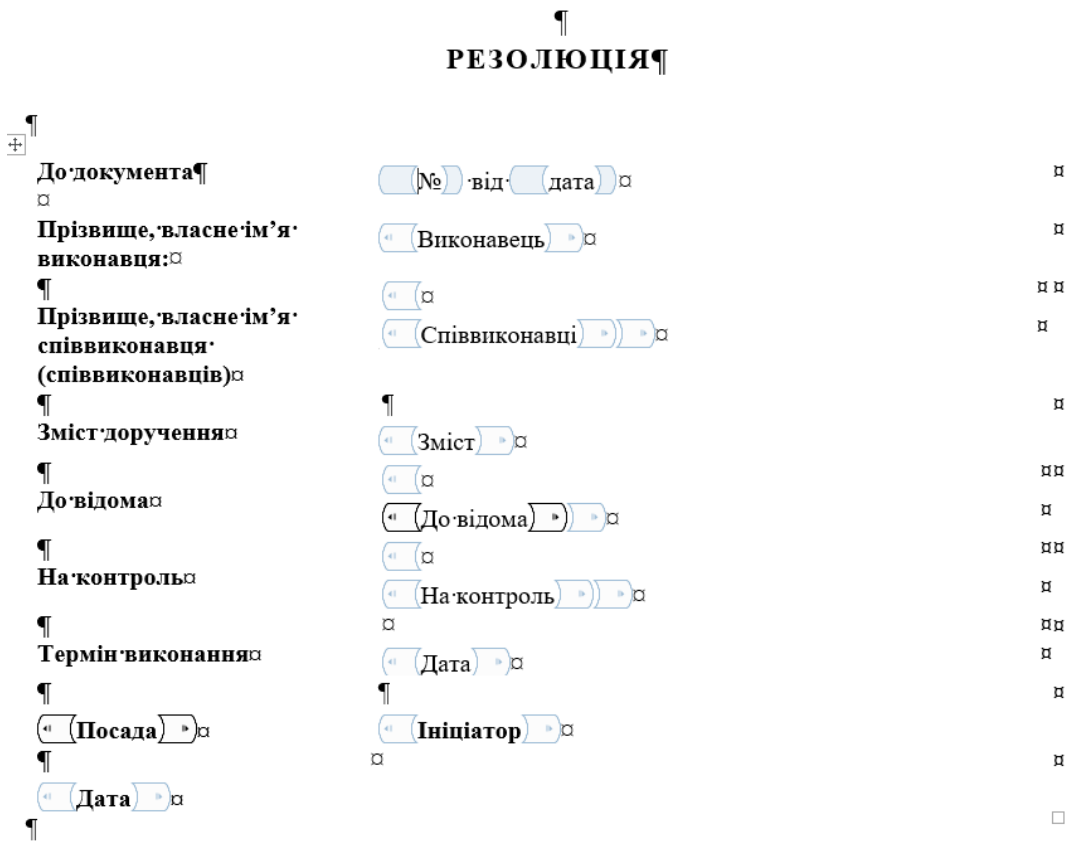
Необхідно дослідити та оглянути кожну частини програмного забезпечення. Вибрати архітектуру для сканеру, та спосіб передачі даних закодованого зображення. Вибрати алгоритм та інтерфейс створення шаблонів. А також дослідити та розглянути правила відмінювання антропонімів.

2.1 Огляд алгоритму створення шаблонів

Для створення шаблонів використовується відкрита бібліотека openXML та її можливості обробки файлів формату docx. Сам алгоритм базується на тому, що за допомогою функції він шукає тег зі своїм певним ідентифікатором. Тег має своє тіло та стилі, що задаються при створенні початкового шаблону без підставлених значень. Всі данні оброблюються певними callback функціями перед тим як подаються у сам двигун. Дані можуть бути статичними, векторними, або малюнками. Статичні дані мають бінарний тип, а векторні дані зберігаються у таблиці формату csv.

Команди на двигун подаються через командну строку. Перший з аргументів це вхідний файл – заготовлений шаблон(рис. 2.1.), другий вихідний файл, третій – аргументи змінних, що в подальшому буде підставлено у вхідний файл.

Також шаблонізатор містить конфіг, що дозволить легко змінювати та налаштовувати функції під кожний шаблон. Це дає можливість легко адмініструвати програмне забезпечення під кожну систему.



¶
РЕЗОЛЮЦІЯ¶

¶		
До документа¶	№ від дата	
Прізвище, власне ім'я виконавця:	Виконавець	
¶		
Прізвище, власне ім'я співвиконавця (співвиконавців)	Співвиконавці	
¶	¶	
Зміст доручення	Зміст	
¶		
До відома	До відома	
¶		
На контроль	На контроль	
¶		
Термін виконання	Дата	
¶	¶	
Посада	**Ініціатор**	
¶		
Дата		
¶		

Рисунок 2.1 - Приклад шаблону документа

2.2 Огляд алгоритму відмінювання антропонімів

Для відмінювання антропонімів було оглянуто та досліджено близько 100 тисяч прізвищ, що підпадають під 100 правил з винятками. Завдяки пріоритетності правил, є можливість відмінювати незмінні жіночі прізвища та прізвища іноземного походження. Для відмінювання потрібно декілька даних, а саме:

- антропонім;
- тип антропоніму(прізвище, ім'я чи патронім);
- рід;
- відмінок(необов'язково вказувати).

Відмінювання відбувається зміною суфіксальної частини[8], або закінчення, що дозволяє швидко оптимізувати сам процес відмінювання, шляхом зіставлення зі зразком. Так як з історії України прізвища походять з

часів козацтва(центральна та східна частина України), а саме реєстр війська Запорізького 1649 року[7], для західного регіону – перші поземельні кадастри Галичини 1785-1788 рр. та 1818-1820 рр., то існує обмежена кількість суфіксів які використовуються у прізвищах [6].

Перелік голосних: а, е, є, и, і, ї, о, у, ю, я

Перелік приголосних: б, в, г, ґ, д, дз, дж, ж, з, к, л, м, н, п, р, с, т, ф, х, ц, ч, ш, щ

Перелік правил виглядає наступним чином:

- прізвища другої відміни, чоловічого роду, м'якої групи на **голосну + вень**;
- прізвища другої відміни, чоловічого роду, м'якої групи на **голосну + лець**;
- прізвища другої відміни, чоловічого роду, твердої групи на **кіт**;
- прізвища другої відміни, чоловічого роду, твердої групи на **піт**;
- прізвища другої відміни, чоловічого роду, твердої групи на **кріт**;
- прізвища другої відміни, чоловічого роду, твердої групи на **пліт**;
- прізвища другої відміни, чоловічого роду, твердої групи на **дріт**;
- прізвища другої відміни, чоловічого роду, твердої групи на **живіт**;
- прізвища другої відміни, чоловічого роду, твердої групи на **грім**;
- прізвища другої відміни, чоловічого роду, твердої групи на **дзвін**;
- прізвища другої відміни, чоловічого роду, твердої групи на **хрін**;
- прізвища другої відміни, чоловічого роду, твердої групи на **батіг**;
- прізвища другої відміни, чоловічого роду, твердої групи на **сокіл**;
- прізвища другої відміни, чоловічого роду, м'якої групи на **кріль**;
- прізвища другої відміни, чоловічого роду, м'якої групи на **голосна + приголосна + ець**;
- прізвища другої відміни, чоловічого роду, м'якої групи на **голосна + приголосна + ень**;

- прізвища другої відміни, чоловічого роду, м'якої групи на **приголосна + приголосна + ець**;
- прізвища другої відміни, чоловічого роду, м'якої групи на **приголосна + приголосна + ень**;
- прізвища другої відміни, чоловічого роду, твердої групи на **рід**;
- прізвища другої відміни, чоловічого роду, твердої групи на **плід**;
- прізвища другої відміни, чоловічого роду, твердої групи на **ніс**;
- прізвища другої відміни, чоловічого роду, твердої групи на **зір**;
- прізвища другої відміни, чоловічого роду, твердої групи на **явір**;
- прізвища другої відміни, чоловічого роду, твердої групи на **орел**;
- прізвища другої відміни, чоловічого роду, твердої групи на **якір**;
- прізвища другої відміни, чоловічого роду, м'якої групи на **бідь**;
- прізвища другої відміни, чоловічого роду, м'якої групи на **мінь**;
- прізвища другої відміни, чоловічого роду, твердої групи на **сіль**;
- прізвища другої відміни, чоловічого роду, твердої групи на **кінь**;
- прізвища другої відміни, чоловічого роду, твердої групи на **куліш**;
- прізвища жіночого роду на **приголосна + ька**;
- прізвища жіночого роду на **ова**;
- прізвища жіночого роду на **іна**;
- прізвища другої відміни, чоловічого роду, м'якої групи на **госна + єць**;
- прізвища другої відміни, чоловічого роду, твердої групи на **віз**;
- прізвища другої відміни, чоловічого роду, твердої групи на **ків**;
- прізвища другої відміни, чоловічого роду, твердої групи на **кіп**;
- прізвища другої відміни, чоловічого роду, твердої групи на **ріп**;
- прізвища другої відміни, чоловічого роду, твердої групи на **піп**;
- прізвища другої відміни, чоловічого роду, твердої групи на **ніс**;
- прізвища другої відміни, чоловічого роду, твердої групи на **віл**;
- прізвища другої відміни, чоловічого роду, твердої групи на **ріг**;

- прізвища другої відміни, чоловічого роду, твердої групи на **хід**;
- прізвища другої відміни, жіночого роду, м'якої групи на **ель**;
- прізвища другої відміни, чоловічого роду, твердої групи на **ишин**;
- прізвища другої відміни, чоловічого роду, твердої групи на **аго**;
- прізвища другої відміни, чоловічого роду, твердої групи на **ово**;
- прізвища другої відміни, чоловічого роду, твердої групи на **тер**;
- прізвища першої відміни, чоловічого роду, твердої групи на **світ**;
- чоловічі імена на **вір**;
- чоловічі імена на **дір**;
- прізвища другої відміни, чоловічого роду, твердої групи на **га**;
- прізвища другої відміни, чоловічого роду, твердої групи на **кіш**;
- прізвища другої відміни, чоловічого роду, м'якої групи на **ьо**;
- прізвища другої відміни, чоловічого роду, твердої групи на **ен**;
- прізвища другої відміни, чоловічого роду, твердої групи на **ев**;
- прізвища другої відміни, чоловічого роду, твердої групи на **пес**;
- прізвища другої відміни, чоловічого роду, твердої групи на **ко**;
- прізвища другої відміни, чоловічого роду, твердої групи на **ий**;
- прізвища другої відміни, чоловічого роду, твердої групи на **йо**;
- прізвища другої відміни, чоловічого роду, твердої групи на **бо**;
- прізвища чоловічого роду на **ой**;
- прізвища чоловічого роду на **ий**;
- прізвища чоловічого роду на **ій**;
- прізвища жіночого роду на **приголосна + а**;
- прізвища жіночого роду на **приголосна + я**;
- прізвища жіночого роду на **ая**;
- прізвища жіночого роду на **яя**;
- патроніми та прізвища чоловічого роду на **ич**;
- патроніми жіночого роду на **на**;
- жіночі імена на **приголосна + я**;

- чоловічі імена на **ня**;
- прізвища другої відміни, чоловічого роду, твердої групи на **ов**;
- прізвища другої відміни, чоловічого роду, твердої групи на **ів**;
- прізвища другої відміни, чоловічого роду, твердої групи на **яр**;
- прізвища другої відміни, жіночого роду, твердої групи на **приголосну**;
- прізвища другої відміни, жіночого роду, м'якої групи на **ь**;
- прізвища другої відміни, жіночого роду, м'якої групи на **о**;
- прізвища другої відміни, чоловічого роду, твердої групи на **ок**;
- прізвища другої відміни, чоловічого роду, твердої групи на **е + приголосна**;
- прізвища другої відміни, чоловічого роду, м'якої групи на **приголосна + я**;
- імена чоловічого, жіночого роду на голосна + **я**;
- імена чоловічого, жіночого роду на **ья**;
- імена чоловічого, жіночого роду на **йя**;
- прізвища другої відміни, чоловічого роду, мішаної групи на **дж**;
- прізвища другої відміни, чоловічого роду, мішаної групи на **ж**;
- прізвища другої відміни, чоловічого роду, мішаної групи на **ш**;
- прізвища другої відміни, чоловічого роду, мішаної групи на **ч**;
- прізвища другої відміни, чоловічого роду, твердої групи на **к**;
- прізвища другої відміни, чоловічого роду, твердої групи на **х**;
- прізвища першої відміни, чоловічого, жіночого роду, твердої групи на **ха**;
- прізвища першої відміни, чоловічого, жіночого роду, твердої групи на **ка**;
- імена першої відміни, чоловічого роду, твердої групи на **приголосна + о**;
- імена першої відміни, жіночого роду, твердої групи на **г**;

- імена першої відміни, жіночого роду, твердої групи на **ґ**;
- прізвища першої відміни, чоловічого роду, твердої групи на **ле**;
- прізвища першої відміни, чоловічого, жіночого роду, мішаної групи на **а**;
- імена другої відміни, чоловічого роду, твердої групи на **приголосну**;
- імена першої відміни, чоловічого, жіночого роду, твердої групи на **а**;
- імена другої відміни, чоловічого роду, м'якої групи на **й**;
- прізвища другої відміни, чоловічого роду, м'якої групи на **ь**;
- жіночі імена на **'я**.

Для кожного правила-взірця є відповідність зміни суфіксів відмінків, що дозволяє заміною другої половини прізвища - відмінювати його.

Також присутні винятки для імен та прізвищ. Вийнятки виглядають наступним чином:

Імена: Любов, Федір, Сидір, Ігор, Лазар

Прізвища: Суддя, Рілля, Сіль

Основними критеріями для відмінювання прізвищ, особових імен та патронімів є рід та частина антропоніму що відмінюється, наприклад, ім’я. Для програмного забезпечення відповідно потрібні правила з патернами на кожний рід, з якими буде зіставлення та подальше відмінювання. Також для правил необхідний пріоритет у списку, для правильності зіставлення. Їх таблиця виглядає наступним чином(табл. 2.1.):

Таблиця 2.1 - Поля та їх значення для правил

Поле	**Значення**
ПІБ	Антропонім (вводиться користувачем)
Рід	Чоловічий або жіночий (вводиться користувачем)
Тип	Прізвище, патронім (вводиться користувачем)
Патерн та зміна на відповідний рід	Зразок зіставлення та подальша заміна
Пріоритет	Певне число

Перевага алгоритму полягає у швидкості відмінювання, а також непотрібні високі вимоги для обчислювальних потужностей.

2.3 Огляд архітектури сканування

Архітектура для сканування буде зроблена за допомогою Twain, адже якість цього рішення залежить від того, що більшість сканерів підтримують роботу з протоколом twain[4]

Сканування відбувається трьома можливими елементами: додаток, ресурс та ресурсний менеджер[5]. Архітектура ж складається з чотирьох рівнів:

- рівень програми;
- протокольний рівень;
- рівень збору;
- рівень пристрою.

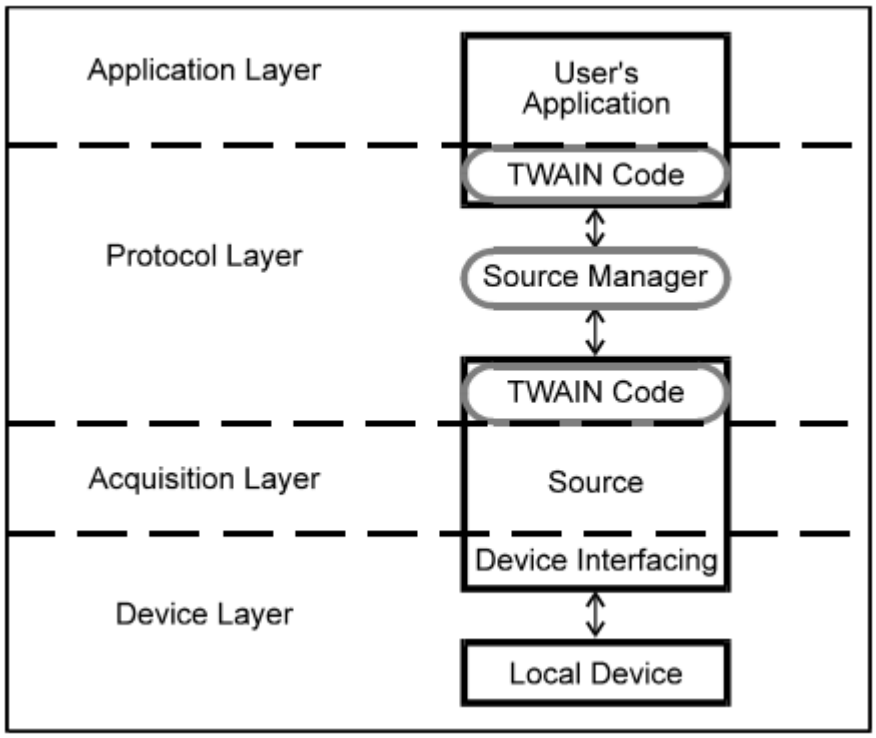


Рисунок 2.2 - Рівні архітектури TWAIN

Рівень програми – рівень програми користувача. На цьому рівні через користувацький інтерфейс надається доступ до функцій Twain.

Протокольний рівень – синтаксис який використовує twain, через який компонуються інструкції та повідомлення для точної передачі даних.

Рівень збору – представляє собою фізичний або логічний пристрій, що має власний інтерфейс.

Рівень пристрою – низькорівневі драйвери пристроїв, що перетворюють програмні команди в апаратні команди та дії.

Перевага даної архітектури полягає у тому, що архітектура дозволяє користуватися функціями переферійних пристроїв зчитування, в даному випадку сканерам, не залишаючи систему(основне програмне забезпечення). Це дозволяє економити час, а також не потрібно підтримувати драйвери для переферійних пристроїв. Також дозволяє самому визначити можливості та функції збору зображень.

Висновки до розділу

У даному розділі було розглянуто основні частини програмного забезпечення їхня архітектура та основні моменти функціонування. Розглянуто правила для відмінювання антропонімів а також було описано переваги обраних технологій.

3 ОПИС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Написання програмного забезпечення для системи обробки електронних документів буде проходити в декілька етапів. Перше це проектування програмного забезпечення, опис базових конструкцій, архітектури, та базових програмних засобів. Далі це саме написання шаблонізатору, сканеру та бібліотеки відмінювання. В кінці написання тестів та тестової інтеграції в примітивний інтерфейс з використанням простої форми для виявлення неточностей та недоліків програмного забезпечення.

Саме програмування в функціональному стилі дозволить з легкістю налаштувати логіку виконання записаних у коді дій.

3.1 Основні мови програмування та бібліотеки

Перед тим як перейти до розробки програмного забезпечення потрібно описати базові конструкції, вибрати архітектуру та базові програмні засоби. Це забезпечить стабільність та адаптивність програмного забезпечення для системи. Для цього потрібно визначити мови програмування та бібліотеки, що можуть працювати з конкурентними та розподіленими системами. Бази даних, які будуть адаптивні до різних таблиць з різними даними. А також протоколи, що зроблять безпечним передавання даних між системою та клієнтом.

3.1.1 Основні мови програмування

За основу клієнто-серверної частини будуть взяті мова програмування така як Erlang , так як у цієї мови висока відмовостійкість та вона підходить для розподілених систем. Для написання конвертації та сканування за основу взятий C#, а для взаємодії між сервером та сервісами F#. Функціональні мови програмування дозволяють програмісту використовувати основну частину

часу на написання бізнес-логіки, а також дозволяють зменшити кількість файлів у проекті та кількість коду, що зменшує час та складність розгортання системи на різних машинах.

3.1.1.1 Erlang

Erlang - це декларативна мова для програмування конкурентних та розподілених систем, розроблена авторами в Лабораторіях Комп'ютерних Наук Ericsson і Ellemtel. Багато примітивів Erlang пропонують рішення проблем, які часто зустрічаються при програмуванні великих конкурентних систем реального часу [10].

Модульна система може структурувати великі програми в керовані блоки. Механізми виявлення помилок дозволяють створювати відмовостійке програмне забезпечення. Примітиви завантаження коду дозволяють змінювати код в працюючій системі без зупинки системи. Erlang має модель паралельності на основі процесів. Користувач має можливість контролювати, які методи виконуються паралельно, а які послідовно. Передача повідомлень між процесами є асинхронною, тобто процес надсилання продовжується, як тільки повідомлення надіслано. Процеси можуть спілкуватися тільки передачею повідомлень. Це призводить до того , що однопоточну програму можна легко змінити для запуску на мультипоточному процесорі, або на багатопроцесній машині. Мова має вбудовані механізми розподіленого програмування. Змінні в Erlang мають одиничне присвоєння, тобто значення змінної не може бути переприсвоєнне. Це спрощує як налагодження так і перетворення програми. Система Erlang має вбудований час, тому у процесах можна вказати скільки очікувати повідомлення, перш ніж продовжувати роботу. Це дозволяє використовувати цю мову для runtime програм, де час відгуку становить мілісекунди. Програми є повністю функціональними, і

вибір функцій здійснюється шляхом pattern matching, це призводить до мінімального коду з та розміру програмних файлів.

Безпека є досить важлива складова в таких системах. У мові існує три конструкції для виявлення помилок під час виконання.

Управління пам’яттю. Erlang - це символьна мова програмування тому garbage collection відбувається runtime. Розподілення пам'яті є автоматичним, і за концепцією мови програмування помилки управління пам'яттю не виникають, так як спільної пам'яті не має, а спілкування відбувається шляхом асинхронної передачі повідомлень між процесами.

Erlang може легко викликати або використовувати програми, написані іншими мовами програмування. Вони можуть бути пов’язані з системою таким чином, що вони здаються програмісту так, ніби вони написані в Erlang. Після порівняльного аналізу всіх існуючих сучасних веб-фреймворків, побудованих з використанням функціональних мов, Erlang став переможцем.

Erlang - досить стара мова, створена у 1980-х. Його метою було вирішити проблему надійного перемикання телекомунікацій. Erlang надав інструменти для створення коду, який є дуже масштабованим, одночасним, з неймовірною тривалістю роботи та стійкістю до несправностей.

3.1.1.2 Elixir

Elixir - це функціональна, динамічно типізована мова, яка побудована поверх BEAM, віртуальної машини Erlang, і компілюється до байтового коду Erlang. Elixir має доступ до всіх інструментів паралельності, до яких має доступ Erlang, що робить його однією з найпотужніших сучасних мов для побудови масштабованих розподілених систем [11].

Від Erlang Elixir використовує платформу OTP, що дозволяє програмісту більшу частину часу витрачати на реалізацію бізнес-логіки.

Erlang має подібний до Ruby синтаксис, а Erlang VM дає продуктивність та конкурентність.

3.1.1.3 F#

Мультипарадигмова мова програмування F# з сімейства dotNET мов, дозволяє писати функціональні програми використовуючі бібліотеки та середу використання dotNET. Йде як доповнення до імперативного та об'єктно орієнтованого програмування.

3.1.1.4 JavaScript, CSS, HTML

Javascript – інтерпритована мова програмування, що використовується для клієнтської частини, а саме виконання певної логіки на стороні клієнта.

HTML – мова розмежування, що визначає структуру вебконтенту, HyperText використовується для з’єднання сторінок між собою, як в межах сайту, так і поза межами

CSS – мова таблиць стилів, що описує зовнішній вигляд сторінки, яка в свою чергу написана на HTML або XML.

3.1.2 Основні бібліотеки та фреймворки

Багато бібліотек та фреймворків мають широкий вибір протоколів, що стабілізує роботу системи, зробить її відмовостійкою та дозволить безпечно передавати дані між сервером, системою та сервісами.

3.1.2.1 N2O

N2O – бібліотека протоколів обміну повідомлень для WebSocket, MQTT, FTP.

N2O вирішує проблеми в веброзробці, залишаючись стислим і малим за розміром. N2O може створювати офлайн-програми на стороні клієнта, за допомогою концепції серверної платформи Nitrogen. Для цього використовується Erlang JavaScript Parse Transform, що дозволяє запускати Erlang на платформі JavaScript, і виконує взаємодію між JavaScript та Erlang. Тобто надає швидке веб-прототипування, з чистим кодом та простим використанням [2].

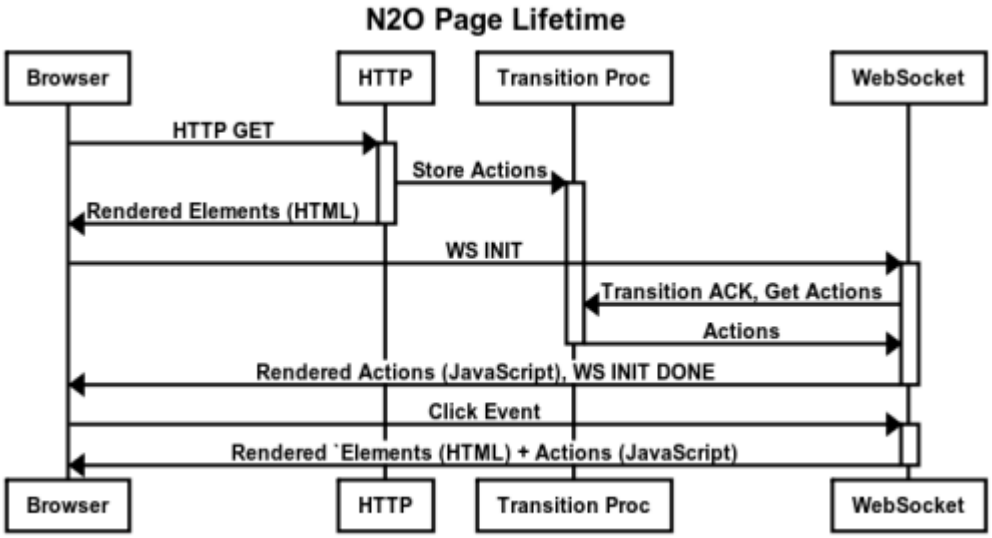


Рисунок 3.1 - Візуалізація сторінки за допомогою N2O

Бібліотека швидко доставляє HTML та відтворює JavaScript лише тоді, коли WebSocket завершив свою роботу.

Термін дії запиту N2O починається з початку процесу HTTP, що обслуговує першу HTML-сторінку. Після цього він вмирає і створюється процес переходу. Потім браузер ініціює підключення WebSocket до подібної кінцевої точки URL-адреси. N2O створює стійкий процес WebSocket, і процес переходу вмирає

3.1.2.2 HEART

Heart - протокол який пінгує клієнта кожні 4-5 сек, що позволяє визначити його присунтість. При повторному чи початковому підключенні клієнт надсилає маркер ініціалізації N2O [2].

3.1.2.3 Nitrogen

Nitrogen дозволяє помічати елементи термінами Erlang, а потім відповідно діяти на них у коді зі сторони серверу, це дозволяє відслідковувати дії користувача. Модель же побудована на узгодженні шаблонів Erlang. Обробка дій користувача є функцією, яка визначає теги HTML із записів Erlang, перевіряючи тип компіляції [2].

Елементи Nitrogen є примітивними елементами керування інтерфейсом, які використовуються для побудови сторінки з внутрішнім DSL Erlang. Взаємодія між сервером та клієнтом відбувається по вебсокету, тому не потрібно використовувати REST. Інтерфейс повністю будується на Erlang, з малою частиною JavaScript або HTML. Це дозволяє повністю адаптувати візуальну складову, а також надає можливість створювати власні елементи.

3.1.2.4 NITRO

Nitro складається з трьох повідомлень: pickle, flush та direct. Протокол дозволяє передавати данні по незахищенному каналу, передаючи тільки значення, шифруючи інформацію про події [2].

3.1.2.5 bert

Модуль bert.js забезпечує кодер/декодер JavaScript для External Term Format, що використовується в протоколі розповсюдження Erlang. Це означає, що програми JavaScript розмовляють з Erlang нативно [2].

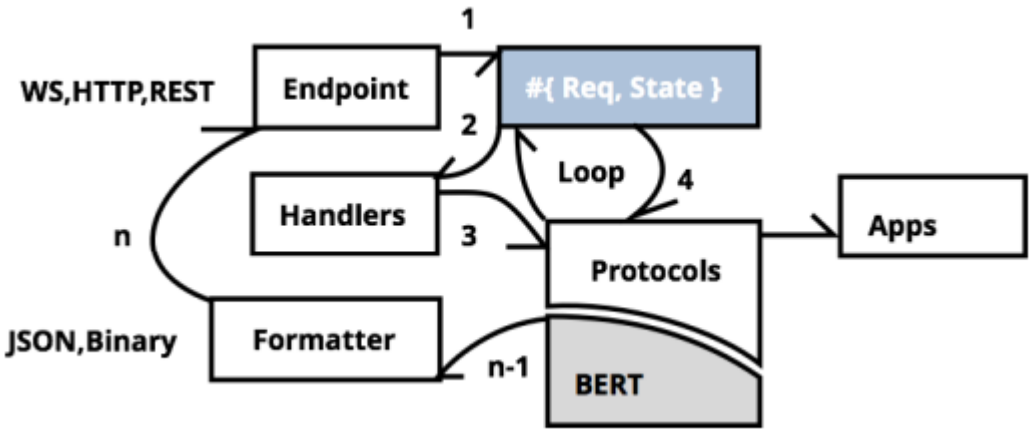


Рисунок 3.2 - Конвеєр передачі повідомлень

Цей модуль використовує DataView та засоби доступу до слів getUint8, getUint16, getUint32, що є найшвидшим способом роботи з бінарними файлами в JavaScript.

3.1.2.6 cowboy

Cowboy - це HTTP-сервер для Erlang OTP з підтримкою протоколів HTTP 1.1, HTTP 2 та Websocket. Cowboy прагне забезпечити повний стек HTTP. Це включає в себе використання HTTP RFC, а також будь-які безпосередньо пов'язані стандарти, такі як Websocket або Server-Sent Events

Коли надходить запит, Cowboy створює потік, який являє собою набір запитів, відповідей та всіх пов’язаних з ними подій. Код протоколу в Cowboy визначає обробку цих потоків для потокових модулів обробника. При налаштуванні Cowboy можна визначити один або кілька модулів, які

прийматимуть усі події, пов'язані з потоком, включаючи запит, відповідь, тіла, повідомлення про Erlang тощо [2].

3.1.2.7 KVS

KVS - це абстракція Erlang над різними власними ключовими значеннями баз даних Erlang. Його мета-схема включає лише поняття ітераторів. Усі операції запису в списоку серіалізовані, використовуючи єдиний процес Erlang для забезпечення послідовності

Модуль KVS забезпечує інтерфейс рівня користувача для консольних команд. Він має функції виявлення, обробки даних та пошуку. KVS обробляє налаштовуванні резервні копії часу виконання для кожної підтримуваної бази даних [1].

3.1.2.8 Mnesia

Mnesia побудована поверх ETS та DETS, щоб додати багато функціональних можливостей у ці дві бази даних. В основному він містить речі, які багато розробників можуть закінчити писати самостійно, якщо їм потрібно буде інтенсивно їх використовувати. Особливості включають можливість автоматичного запису як в ETS, так і в DETS, щоб обидва мали стійкість DETS та продуктивність ETS, або можливість автоматичного копіювання бази даних на багато різних вузлів Erlang. Таблиці ETS - це ефективна база даних у пам’яті, що входить до складу віртуальної машини Erlang. Як правило, вони швидкі, і це досить простий спосіб для програмістів Erlang оптимізувати частину свого коду, коли його частини стають занадто повільними [1].

3.1.2.9 RocksDB

RocksDB - це механізм зберігання даних з інтерфейсом ключ-значення, де ключі та значення є довільними потоками байтів. Він може адаптуватися до різних виробничих середовищ, включаючи чисту пам'ять, флеш-пам'ять, жорсткі диски або віддалене сховище. Там, де RocksDB не може автоматично адаптуватися, надаються надзвичайно гнучкі параметри конфігурації, що дозволяють користувачам налаштовувати їх під них [1].

3.1.2.10 OpenXML

Бібліотека dotNet що дозволяє працювати з документами, яким відповідає специфікація Office Open XML. Дозволяє перетворювати ZIP та XML в елементні схеми, що полегшує написання коду для виконання стандартних задач та складних операцій.

3.2 Програмна реалізація шаблонізатору

Для шаблонізатору напишемо інтепритатор який приймає строку аргументів.

```
CRM.QR.execute('priv/docx/docx -vars #{Enum.join(data, " , ")} -in "priv/docx/in#{guid}.docx"' ++
               '-out "priv/#{dir}/#{guid}.docx"', err))
```

Рисунок 3.3 - Приклад компоновки команди для генератора шаблонів

- priv/docx/docx – шлях до exe файлу генератора;
- vars data – перелік змінних з обробленими значеннями;
- in file – шлях до шаблону;
- out file – шлях для створення згенерованого документу.

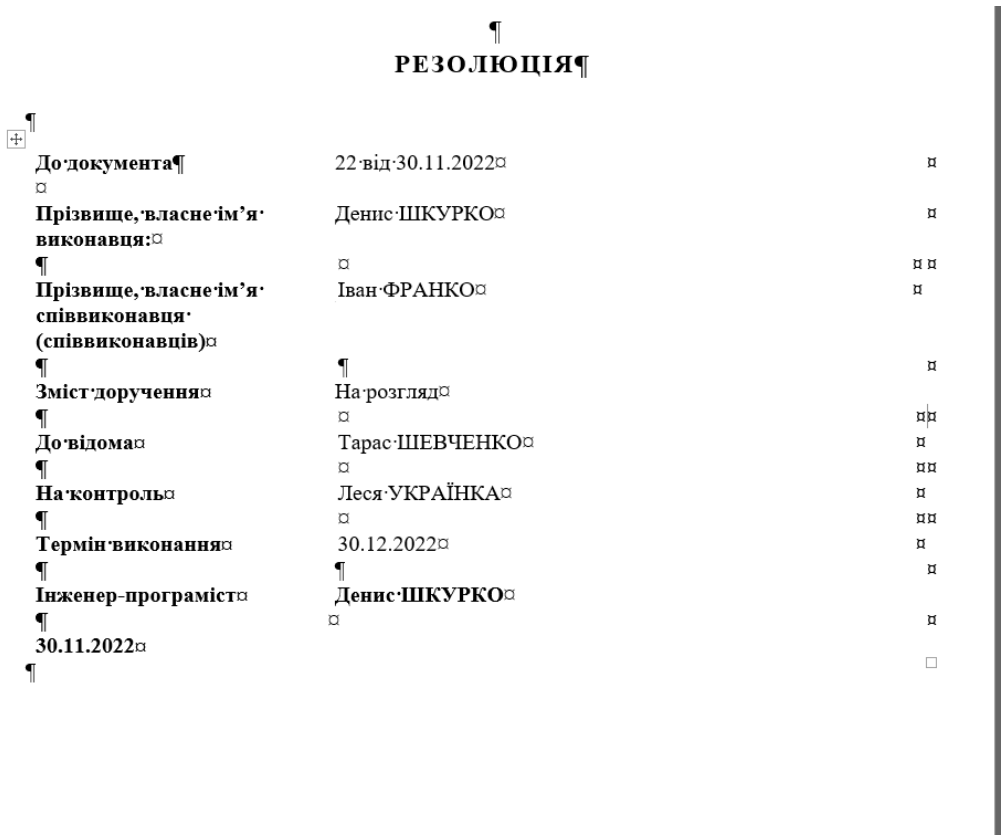
Рекорд змінної виглядає наступним чином(рис. 3.4.)

Рисунок 3.4 - Record змінної

- id – назва змінної;
- record – прототип рекорду;
- type – тип поля, може бути скалярним, або векторним;
- main – змінна для callback;
- tableName – назва таблиці;
- tableGuid – унікальна назва для csv з даними для векторного типу;
- field – ключ прототипу;
- value – дефолтне значення або змінна для таблиці;
- callback – функції для послідовного виклику перетворення значень;
- options – додаткові опції, конфігурація.

Для оброблення значень в таблицю передаються посилання на функції, що виконуватимуться при обробці значення перед підстановкою у шаблон.

Callback – можуть бути любі функції обробки, наприклад відмінювання



¶
РЕЗОЛЮЦІЯ¶

¶

До документа¶	22 від 30.11.2022¤	¤
¤		
Прізвище, власне ім'я виконавця:¤	Денис ШКУРКО¤	¤
¶	¤	¤ ¤
Прізвище, власне ім'я співвиконавця (співвиконавців)¤	Іван ФРАНКО¤	¤
¶	¶	¤
Зміст доручення¤	На розгляд¤	
¶	¤	¤ ¤
До відома¤	Тарас ШЕВЧЕНКО¤	¤
¶	¤	¤ ¤
На контроль¤	Леся УКРАЇНКА¤	¤
¶	¤	¤ ¤
Термін виконання¤	30.12.2022¤	¤
¶	¶	¤
Інженер-програміст¤	**Денис ШКУРКО**¤	
¶	¤	¤
30.11.2022¤		
¶		□

Рисунок 3.5 - Приклад заповнення шаблону

3.3 Програмна реалізація алгоритму відмінювання

Робота алгоритму діє за наступною послідовністю дій(рис. 1):

- користувач вводить антропонім, його тип та рід;
- відбувається зіставлення зі взірцем та вибір правила;
- відмінювання введеного користувачем антропоніма;
- ввиведення провідмінованого антропоніму.

Рисунок 3.6 - Алгоритм роботи програмного забезпечення відмінювання

Для відмінювання антропонімів було розглянуто близько 100 правил української мови та написано близько 1000 взірців для зіставлення(рис. 3.6.).

	B	C	D	E	F	G	H	I	J	K	L
1	Рід	Патерн	Тип	Називний	Родовий	Давальний	Знахідний	орудний	місцевий	кличний	приорітет
2	Ж	любов	Ім'я	любов	любові	любові	любов'ю	любові	любове	любове	6
3	Ч	федір	Ім'я	федір	федора	федору	федора	федором	федорові	федоре	6
4	Ч	сидор	Ім'я	сидор	сидора	сидору	сидора	сидором	сидорові	сидоре	6
5	ЧМ	суддя	Прізище	суддя	судді	судді	суддю	суддею	судді	суддю	6
6	ЧМ	рілля	Прізвище	рілля	ріллі	ріллі	ріллю	ріллею	ріллі	ріллю	6
7	Ч	(голосна)вень	-	вень	вня	вню / вневі	вня	внем	вневі	вню	6
8	Ч	(голосна)лець	-	лець	льця	льцю	льцеві	льця	льцем	льцю	6
9	Ч	(к|п|кр|пл|др|жив)іт	-	іт	ота	оту/отові	ота	отом	отові/оті	оте	6
10	Ч	()грім	-	грім	грома	грому/громові	грома	громом	громі/громові	громе	6
11	ч	(дзв|хр)ін	-	ін	она	ону	она	оном	оні	оне	6
12	ч	батіг	-	батіг	батога	батогу	батога	батогом	батозі	батогу	6
13	Ч	сокіл	-	сокіл	сокола	соколу	сокола	соколом	соколові	соколе	6
14	Ч	кріль	-	кріль	кроля	кролю	кроля	кролем	кролю	кролю	6
15	Ч	ігор	ім'я	ігор	ігоря	ігорю	ігоря	ігорем	ігорю	ігоре	5
16	Ч	лазар	ім'я	лазар	лазаря	лазарю	лазаря	лазарем	лазарю	лазоре	5
17	Ч	сіль	-	сіль	солі	сіллю	сіль	соллю	солі	соле	5
18	Ч	(а|е|є|и|і|ї|о|у|ю|я)(б|в|г|ґ|д|дз|дж|ж|з|	-	е(ц|н)ь	(ц|н)я	(ц|н)ю	(ц|н)я	(ц|н)ем	(ц|н)ю	(ц|н)е	5
19	Ч	(б|в|г|ґ|д|дз|дж|ж|з|к|л|м|н|п|р|с|т|ф|х	-	(ц|н)ь	(ц|н)я	(ц|н)ю	(ц|н)я	(ц|н)ем	(ц|н)ю	(ц|н)ю	5
20	Ч	рід	-	рід	рода	роду	рода	родом	родові	роде	5
21	Ч	плід	-	плід	плода	плоду	плода	плодом	плодові	плоде	5
22	Ч	(б|в|г|ґ|д|дз|дж|ж|з|й|к|л|м|н|п|р|с|т|ф	-	ніс	носа	носу	носа	носом	носові	носе	5

Рисунок 3.7 - Таблиця патернів та відмінків

Так як зіставлення залежить від пріоритету, то правило з вищим пріоритетом буде відмінювати введений антропонім. Візьмемо прикладом

чоловіче прізвище Камінь. Вказане прізвище підпадає під два правила, але вибране правило буде з вищим пріоритетом:

Перше правило

Rule: прізвище другої відміни чоловічого роду м’якої групи на -мінь

Priority: 5

Pattern: “^мінь”

Replace:

Denominative: “мінь”

Genitive: “меня”

Dative: “меню”

Accusative: “меня”

Albative: “менем”

Locative: “меню”

Vocatile: “меню”

Друге правило

Rule: прізвище другої відміни чоловічого роду м’якої групи на -ь

Priority: 1

Pattern: “ь”

Replace:

Denominative: “ь”

Genitive: “я”

Dative: “ю”

Accusative: “я”

Albative: “ем”

Locative: “еві”

Vocatile: “ю”

3.4 Програмна реалізація сканування

Для сканування для початку потрібно вибрати драйвер пристрою(рис. 3.8.). Це відбувається шляхом ініціалізації TWAIN ынтерфейсу та передачі в нього певного повідомлення: Twain.DatIdentity(TWAIN.DG.CONTROL, TWAIN.MSG.SET, ref identity)

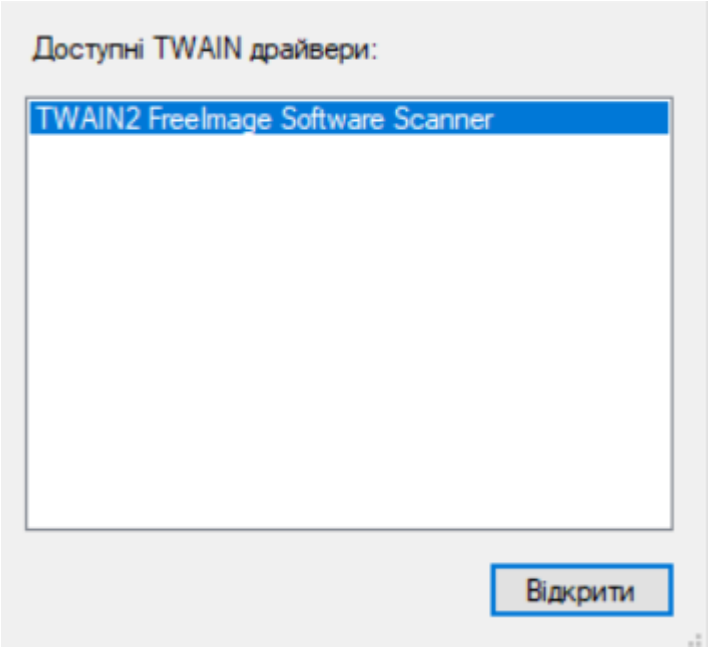


Рисунок 3.8 - Вікно вибору сканера

Далі потрібно вибрати налаштування принтеру(рис. 3.10.), після чого стає доступним сканування. Також є можливість вибору стандартного конфігу. Всі capabilitis описані в документації TWAIN і шляхом спілкування з драйвером периферійного пристрою повертаються можливі налаштування.

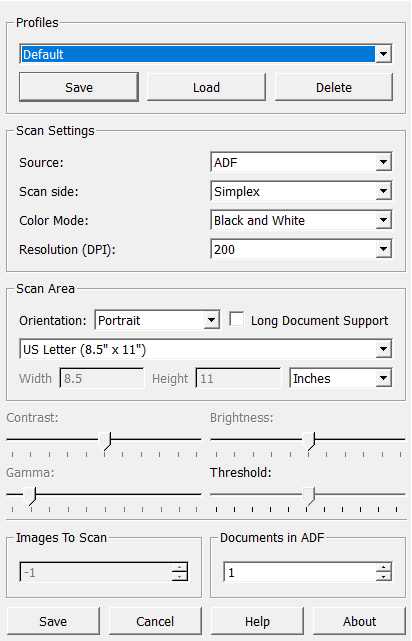


Рисунок 3.9 - Налаштування сканування

Повну діаграму переходу станів можна побачити у додатку.

```
dentity(Default): 3,2,1,ENG,USA,2.1.3 sample release 32bit,2,1,0x40000003,TWAIN Working Group,Software Scan,TWAIN2 FreeImage Software Scanner
canner(1): 3,2,1,ENG,USA,2.1.3 sample release 32bit,2,1,0x40000003,TWAIN Working Group,Software Scan,TWAIN2 FreeImage Software Scanner
t: Untested Windows version 6.2 detected!
penDS: SUCCESS
ferMechanism(Native): SUCCESS
utoFeed(1): SUCCESS
ndicators(1): SUCCESS
ormSetup constructor
etCurrentCap(CAP_CUSTOMDATA): SUCCESS CAP_CUSTOMDSDATA,TWON_ONEVALUE,TWTY_BOOL,TRUE
ondition: False
WAIN UI: [CONTROL] [ENABLEDSUIONLY] CSV: FALSE,FALSE,200984
nableDSM_UI(FALSE,FALSE,200984): SUCCESS
can Callback Entered: False Sate S5 Range 0-0
can Callback IsMsgCloseDsReq True
sMsgCloseDsReq True
ollback to state: S4
can Callback IsMsgCloseDsOk False
ormSetup constructor
etCurrentCap(CAP_CUSTOMDATA): SUCCESS CAP_CUSTOMDSDATA,TWON_ONEVALUE,TWTY_BOOL,TRUE
ondition: False
s: rescaling... to 1600 x 2200
s: converting to TWPT_BW...
can Callback Entered: False Sate S6 Range 0-0
can Callback IsMsgCloseDsReq False
can Callback IsMsgCloseDsOk False
ending End
mageNativeXfer(): XFERDONE
ATIVE GET: {Width=1700, Height=2200} 0 D:\
ile: D:\img-000001.tif
mageInfo: 200,200,1700,2200,1,1,0,0,0,0,0,0,0,0,0,0,TWPT_BW,TWCP_NONE
ND XFER: 0 0
rocessPDF 1-1
canning done: SUCCESS
reatePDF 1-1
```

Рисунок 3.10 - Логи сканування додатку

3.5 Тестування програмного забезпечення

Для тестування скрипту відмінювання антропонімів були написані модульні тести, що для прикладу перевірки використовують дата сет з 1000 прізвищ, це дозволяє перевіряти якість виконання програми при додаванні, або зміненні функціоналу.

У таблиці 3.1 та таблиці 3.2 зображені приклади роботи, очікувані та реальні результати, з якої можна зробити висновок, що усе працює вірно.

Таблиця 3.1 - Тестування сканування

Дія користувача	Очікуваний результат	Реальний результат
Вибір принтеру	Демонструється вікно з вибором сканеру	Демонструється вікно з переліком принтерів та кнопкою обрати
Конфігурація	Демонструється вікно з налаштуванням сканеру	Демонструється вікно з налаштуванням сканеру, а також кнопка стандартного налаштування
Сканування	Відбувається сканування	Відбувається сканування, на екрані з'являється відсканований документ

Таблиця 3.2 – Тестування створення шаблону

Дія користувача	Очікуваний результат	Реальний результат
Вибір шаблону, змінних та їх значення та початок створення шаблону та шляху для збереження результату.	Створення нового файлу зі значеннями підставлених змінних	У папці вказаного шляху з'являється новий файл docx формату із заповненими вказаними значеннями

Висновки до розділу

Отже, було реалізовано програмне забезпечення для сканування, шаблонізатору та скрипт для відмінювання антропонімів. Перевагою цих програм є те, що вони легко налаштовуванні, що дозволяє впровадити їх у будь-яку систему документообігу. Виконано тестування програмного забезпечення.

4. ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РОБОТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Експериментальне дослідження.

Для проведення експерименту було використано персональний комп’ютер з наступними характеристиками:

- операційна: система Ubuntu 20.04LTS;
- процесор: intel core i5-9300H;
- оперативна пам’ять 6 Гб DDR4 та 2 Гб файлу підкачки;
- накопичувач SSD WDC SN520 SDAPNUW-256GB-1002.

Для прототипу системи була використанна база даних rocksdb, та написана веб сторінка. В якості функцій для обробки даних використовується скрипт для відмінювання антропонімів.

Так як швидкість системи залежить від швидкості накопичувача, на зображенні 4.1 наведено його характеристики, де швидкість зчитування 160Mib/s, а швидкість запису 55.6Mib/s, швидкість обробки 43 тисячі операцій зчитування на секунду та 14 тисяч операцій запису на секунду

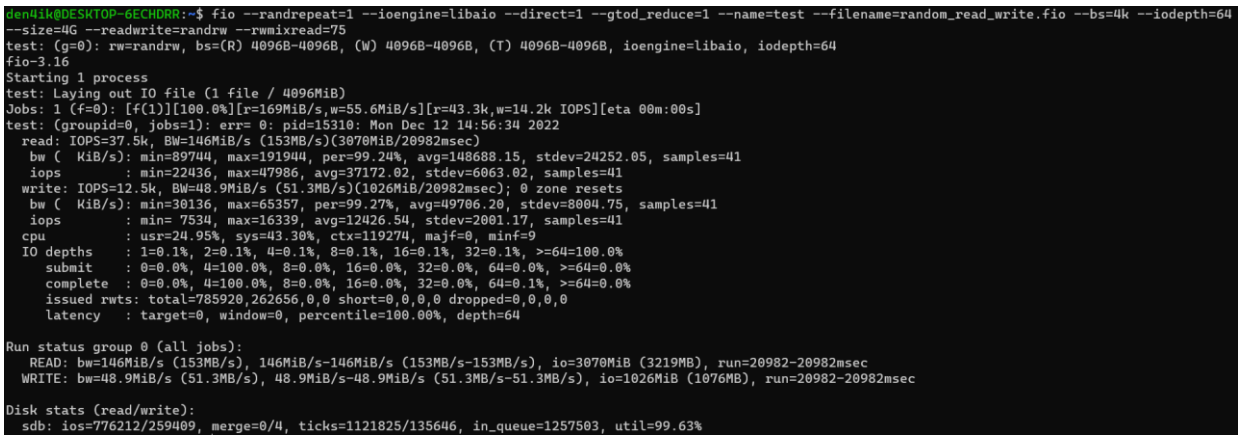


Рисунок 4.1 - Характеристики SSD

Ефективність роботи можна оцінити паралельним створенням декількох шаблонів, це дозволить визначати завантаженість системи та швидкість роботи програмного забезпечення

Результати проведення експерименту зображені на рис 4.2 – 4.5.

Рисунок 4.2 - Швидкість 1 одночасного створення у мкс

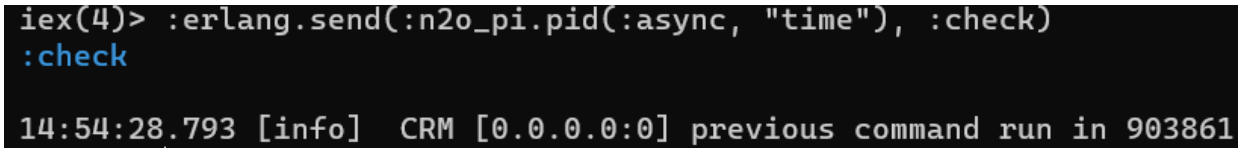


Рисунок 4.3 - Швидкість 10 одночасних створень у мкс

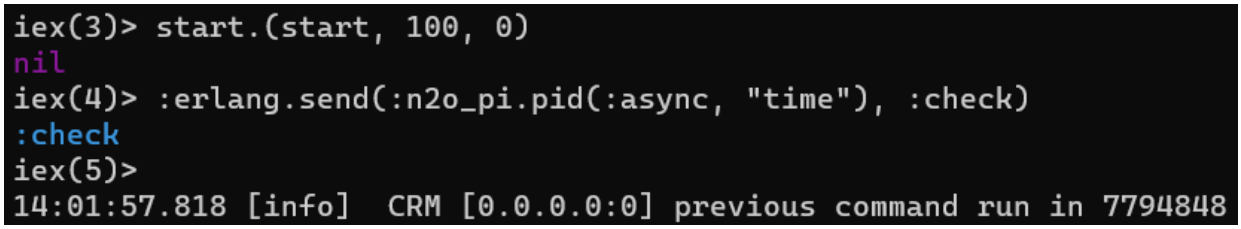


Рисунок 4.4 - Швидкість 100 одночасних створень у мкс

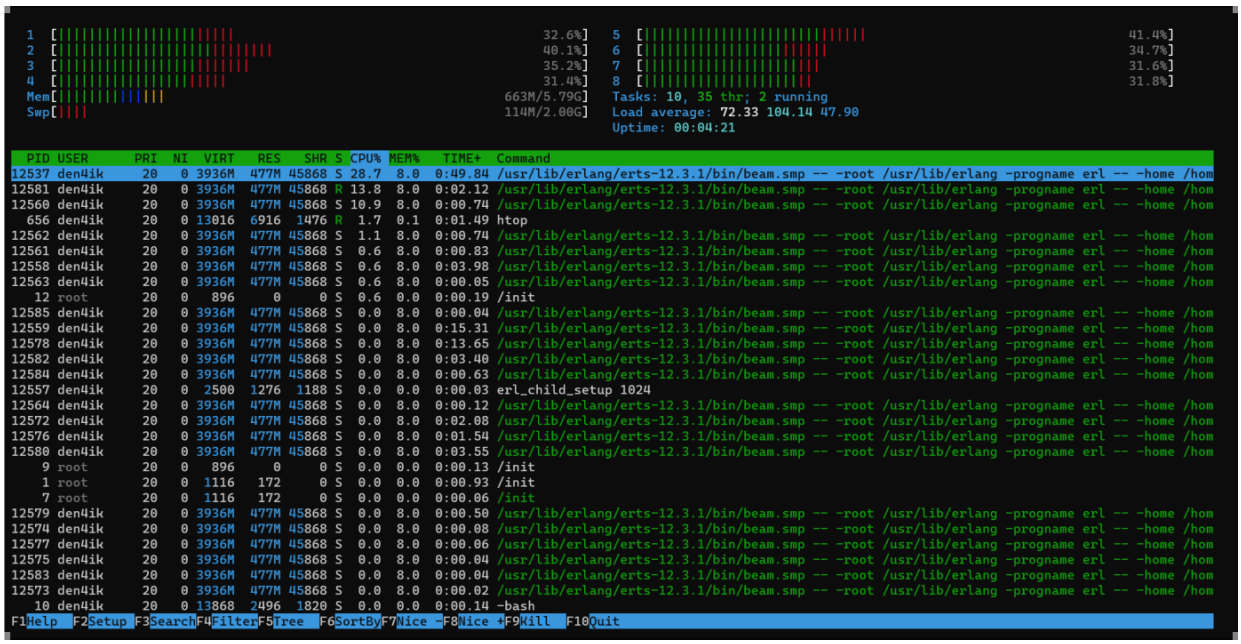


Рисунок 4.5 - Завантаженість серверу при 100 одночасних створень

З результату експерименту помітно, що час одночасного створення шаблонів для одного шаблону склав 419 мс, для десяти – 903мс, для 100 – 7795мс. Завантаженість процесору не перевищувала 60% на кожне ядро, при 100 одночасних створеннях. Використання оперативної пам'яті до 2 гб.

Порівняльна характеристика наведена у таблиці 4.1

Таблиця 4.1 - Порівняльна характеристика для різної кількості одночасних запусків створення шаблонів з попередньою обробкою даних

	1 процес створення	10 паралельних процесів створення	100 паралельних процесів створення
Загальний час, мс	419	903	7795
Завантаженість процесору, %	30	30	60
Використання оперативної пам'яті, гб	1	1	2

Висновки до розділу

Виконано оцінку ефективності запропонованого рішення, що показало час для одночасного створення декількох шаблонів, завантаженість процесору серверу, та використання оперативної пам’яті. Зроблена загальна порівняльна характеристика, що дозволило визначити завантаженість системи та швидкість роботи програмного забезпечення

5 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП ПРОЄКТУ

5.1 Опис ідеї проєкту

По результатам проведеного дослідження було знайдено потребу у додатковому програмному забезпеченні для систем обробки електронних документів, а саме шаблонізатор та програма для сканування. Це дозволяє зменшити обсяг і час роботи компаній, які працюють зі звітністю та документами

Для досягнення мети було розроблено шаблонізатор та програму для сканування, а також скрипт відмінювання антропонімів.

Таблиця 5.1 – Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Створити програмне забезпечення що дозволяє сканувати документи, створювати шаблони docx та відмінювати антропоніми, для систем обробки електронних документів.	Електронний документообіг	Програмне забезпечення не потребує високих обчислювальних потужностей та легко може бути впроваджено у будь яку систему. За рахунок конфігурацій відбувається легке налаштування під систему.
	Системи звітності та статистики	

У таблиці 5.2 проведено порівняльний аналіз основних техніко-економічних характеристик ідеї з товарами-аналогами.

Таблиця 5.2 – Порівняльна характеристика з товарами-аналогами

№	Техніко-економічні характеристики ідеї	Продукція конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	mustache	ETC			
1	Системні вимоги	Низькі	Середні	Середні			+
2	Зручність використання	Висока	Низька	Низька			+
3	Програмні залежності	Низькі	Середні	Високі			+
4	Кросплатформеність	Так	Так	Ні		+	
5	Документація	Ні	Так	Так	+		
6	Підтримка мов для написання скриптів	Будь-яка	Будь-яка	Coffescript, js, node.js		+	
7	Деталізованість середовища	Висока	Висока	Низька		+	
8	Підтримка баз даних	Будь-які	Будь-які	Будь-які		+	

5.2 Технологічний аудит проєкту

Для визначення технологічної здійсненності проєкту потрібно визначити технології та проаналізувати їх доступність, що наведено у таблиці 5.3

Таблиця 5.3 – Технологічна здійсненність ідеї проєкту розробки програмного забезпечення для систем обробки електронних документів

№	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Протокол для сканування документів	Twain	Наявність	Доступність
		SANE	Наявність	Доступність
		WIA	Наявність	Недоступність
Обрана технологія для реалізації технології: Twain				
2	Мова програмування для скрипту відмінювання антропонімів	F#	Наявність	Доступність
		Erlang/Elixir	Наявність	Доступність
Обрана технологія для реалізації технології: Erlang/Elixir				

5.3 Аналіз ринкових можливостей запуску стартап проєкту

Для того щоб проаналізувати ринок, потрібно провести аналіз потенційних постачальників систем документообігу.

Таблиця 5.4 - Попередня характеристика потенційного ринку стартап проєкту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців	3
2	Загальний обсяг продаж, грн/ум.од	432000$
3	Динаміка ринку (якісна оцінка)	зростає
4	Наявність обмежень для входу (вказати характер обмежень)	немає

Кінець таблиці 5.4

5	Специфічні вимоги до стандартизації та сертифікації	ISO-42010 ISO-19510 ISO-20922
6	Середня норма рентабельності в галузі (або по ринку), %	29%

На основі проведеного дослідження є можливість стверджувати про привабливість проєкту для входження на ринку

У таблиці 5.5 наведено характеристику потенційних клієнтів стартап-проєкту, у таблиці 5.6 наведено фактори загроз, у таблиці 5.7 наведено ступеневий аналіз конкуренції на ринку.

Таблиця 5.5 - Характеристика потенційних клієнтів

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у проведені різних потенційних груп	Вимоги споживачів до товару
1	Програмне забезпечення для систем обробки електронних документів	Державні структурні органи управління, приватні та юридичні компанії	Очікують продукт, що відповідає внутрішній організації системи документообігу	Продукт відповідає постанові 55 Кабінету Міністрів України

Таблиця 5.6 - Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Надання невірних даних	Помилкове надання недостовірних даних	Програмне забезпечення має можливість оновити дані після їх редагування

Кінець таблиці 5.6

2	Висока вартість інформаційної системи	Збільшення ціни на систему	Підвищить агресивність конкурентів
3	Економічні складності при розробці	Повна відсутність фінансування	Може порушити фінансове забезпечення компанії

У таблиці 5.8 наведено ступеневий аналіз конкуренції на ринку, а у таблиці 5.9 описано аналіз конкуренції в галузі за М. Портером

Таблиця 5.7 - Фактори можливостей

№	Фактор	Зміст можливостей	Можлива реакція компанії
1	Висока вартість програмного забезпечення	Встановлення високої ціни на програмне забезпечення	Ускладнює вихід на нові ринки
2	Моніторинг потреб	Можливість розширювати діапазон методів попередньої обробки цифрових даних	Розширення діапазону методів цифрових даних

Таблиця 5.8 - Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною
1	Тип конкуренції: Чиста	Багато конкурентів	Кооперація
2	Рівень конкурентної боротьби: державний рівень	Конкуренти у державі	Проводити залучення клієнтів у державі
3	Галузева ознака: Внутрішньо-галузева	Внутрішньогалузева. Конкуренція на ринку ведеться в інформаційній галузі України	Необхідно зосередити зусилля на пошуку конкурентних переваг, які дозволять компанії займати стійкі конкурентні позиції
4	Конкуренція за видами товарів: Товарно-видова	Конкуренція між товарами різного виду	Додавання нового функціоналу

Кінець таблиці 5.8

5	Характер конкурентних переваг: Нецінова	Конкуренція проводиться за рахунок удосконалення якості продукту	Вдосконалення продукту
6	За інтенсивністю: Немарочна	Роль торгової марки незначна	Заохочення клієнтів якістю товару, а не брендом.

Таблиця 5.9 - Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Процес	Висновок
Прямі конкуренти в галузі	Навести перелік прямих конкурентів	Конкурентами можуть виступати постачальники систем документообігу
Потенційні конкуренти	Визначити бар’єри входження в ринок	Бар'єри входу на ринок є порівняно незначними
Постачальники	Визначити фактори сили постачальників	Існує чітка залежність від постачальників в якості та оновленні програмного забезпечення

Проведений аналіз ринкових можливостей запуску стартап проєкту показав, що ринок для входження проєкту є привабливим, потенційними клієнтами можуть бути як державні, так і приватні, юридичні компанії.

Існує фактор загроз, але на противагу фактори можливостей дають оптимістичну оцінку запуску проєкту. Аналіз конкуренції в галузі показав актуальність та доступність застосованих технологій, що доводить – можливості входу високі.

У таблиці 5.10 обґрунтовано фактори конкурентності товару.

Таблиця 5.10 - Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Актуальність товару	Застосування новітніх технологій, що привертають увагу споживачів
2	Висока готовність до кооперації	Відкритість до діалогу
3	Простота використання	Зрозумілість відправки команд
4	Легка можливість впровадження в систему	Легка налаштованість
5	Кросплатформеність	Можливість використовувати на різних платформах

Таблиця 5.11 - Порівняльний аналіз сильних та слабких сторін програмного забезпечення для навчання та тестування самокерованих транспортних засобів

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			-3	-2	-1	0	1	2	3
1	Актуальність товару	19				+			
2	Висока готовність до кооперації	15				+			
3	Простота використання	17		+					
4	Легка можливість впровадження в систему	20	+						
5	Кросплатформеність	20	+						

Отже можна зробити висновок, що розроблене програмне забезпечення має значні переваги над товарами-конкурентами.

У таблицях 5.12 та 5.13 наведено характеристику слабких та сильних сторін продукту, а також альтернатив ринкового впровадження

Таблиця 5.12 – SWOT аналіз стартап-проекту

Сильні сторони (S):	Слабкі сторони (W):
Програмне забезпечення легконалашовуване, досить мало конкурентів на ринку.	Необхідність інвестицій
Можливості (O): - Обробка даних; - Налаштовуваність; - Багатозадачність.	Загрози (T): - Недовіра споживачів.

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Пошук компаній для постачання	Головний ресурс – гроші, даний ресурс – необхідне залучення	6 місяців
2	Впровадження у систему	Головний ресурс – час, даний ресурс - наявний	1 місяць
3	Навчання користуванню	Ресурс – час, наявний	2 тижня

5.4 Розроблення ринкової стратегії проекту

Було проведено дослідження цільових груп споживачів, для яких орієнтовано програмне забезпечення та визначено простоту входу у сегмент

Таблиця 5.14 – Вибір цільових груп потенційних споживачі

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Державні установи	Висока, адже завжди працюють з документами	85% - високий попит	Низька	Середня
2	Юридичні компанії	Середня, адже переважно працюють з документами	50% - середній попит	Висока	Середня
3	Приватні компанії	Низька, адже рідко працюють з документами	5% - низький попит	Низька	Низька
Обрані цільові групи: державні установи та юридичні компанії					

Таблиця 5.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Розвиток продукту та постійне оновлення функціоналу по мірі розвитку технологій	Маркетинг нових ринків	Активний розвиток технологій штучного інтелекту та жага виробити автономний автомобіль буду сприяти розвитку продукту	Стратегії концентрованого зростання

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Ні	Буде шукати нових споживачів	Буде використовувати вже існуючі програмні рішення, але застосовувати та розроблювати нові модулі	Кооперaційна стратегія

Таблиця 5.17 – Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Здатність системи до розширювання	Стратегії диверсифікованого зростання	Програма повинна бути побудована легко налаштовуваною	відкритість до змін гнучкість

Кінець таблиці 5.17

2	Можливість заміни різних платформ, що використовує система	Стратегії диверсифікованого зростання	Програма буде розроблена з позиції принципів проектування, що дозволять клієнтам застосовувати різні платформи для інтеграції вже з існуючими системами	кросплатформеність економічна вигода залучення більшої кількості клієнтів

Відповідно до проведеного аналізу можна зробити висновок, що стратегія розвитку – стратегія концентрованого зростання. Базова стратегія конкурентної поведінки – кооперaційна стратегія

5.5 Розроблення маркетингової програми стартап-проекту

Розроблення маркетингової програми передбачає визначення ключових переваг концепції потенційного товару, формування системи збуту та вибір концепції маркетингових комунікацій та опис трьох рівнів моделі товару, визначення меж встановлення ціни. Результати представлено в таблицях нижче.

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Сканування	Налаштовуване сканування із видимістю мережевих сканерів	Висока задіяність технологій та зручність

Кінець таблиці 5.18

2	Шаблонізатор	Шаблонізатор, який з легкістю зробить шаблон	Налаштовуваміть та висока сумісність з будь-якою системою
3	Відмінювання антропонімів	Скрипт, який провідміняє ПІБ як українського, так і іноземного походження	Зручність, швидкість, точність

Таблиця 5.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
1. Товар за задумом	Програмне забезпечення для систем обробки електронних документів		
2. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	1. Налаштовуваність 2. Функціональність 3. Кросплатформеність	Нм М Нм	Тх/Е/Ор Тл Тх
	Якість безпеки відповідає стандарту X.509		
	Пакування: Характеристики продукту будуть оформлено в офіційній документації, що буде доступна на GitHub		
3. Товар із підкріплення м	До продажу: Програмний додаток		
	Після продажу: постійне розширення функціоналу, підтримка продукту		

*М/Нм – монотонні/немонотонні; *Вр/Тх/Тл/Е/Ор – вартісні/ технічні/ технологічні/ ергономічні/ органолептичні

Таблиця 5.20 – Визначення меж встановлення ціни

Рівень цін на товари	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на
750$	1000$	Високий для можливої комерційної версії продукту	500$ - 1000$

Таблиця 5.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Цільові клієнти можуть долучитися до розвитку програмного забезпечення або його у розробника	- аналіз ринку - дослідження ринків для розширення - аналіз законодавчих норм	Багатоканальний розподіл	Українські компанії, державні установи

Таблиця 5.22 – Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Клієнти зацікавлені особи, що відвідують масові заходи присвячені технологіям	Конференції, виставки, саміти	Налаштовуваміть, легкість впровадження у систему	Залучити зацікавлених осіб	Розум перемагає силу

Таким чином доведено, що продукт задовольняє вимоги, було описано товар у реальному виконанні та підкріплені.

Висновки до розділу

В четвертому розділі було оглянуто ідею стартап-проекту. проведено порівняльний аналіз основних техніко-економічних характеристик ідеї з товарами-аналогами. Було проведено дослідження цільових груп споживачів. наведено характеристику потенційних клієнтів стартап-проєкту та розроблення маркетингової програми

ВИСНОВКИ

У ході виконання магістерської дисертації було написано програмне забезпечення для сканування і створення шаблонів, з використанням змінної конфігурації, що дозволяє з легкістю налаштовувати ці продукти для будь-якої системи документообігу.

Були проаналізовані алгоритми для підстановки значень у документи, що дозволили написати програму для створення шаблонів. Були проаналізовані алгоритми відмінювання антропонімів, досліджені правила суфіксального відмінювання прізвищ, імен, по-батькові для подальшого зіставлення зі взірцем Проаналізовані алгоритми сканування, та вибрана зручна архітектура для написання програми сканування.

Для виконання поставлених задач були використані чотири мови програмування, а саме F#, C#, Erlang та Elixir. Описано та проаналізовано стек технологій зручний для написання програмного забезпечення. Показана сама програмна реалізація та виконана її оцінка ефективності.

Був проведений маркетинговий аналіз стартап-проєкту та розроблено ринкову стратегію, детально описана ідея, технологічний аудит проєкту.

Результати роботи над магістерською дисертацією опубліковані у одній статті.

Наукова новизна одержаних результатів магістерської дисертації здобула подальший розвиток, а саме:

- створено конфігуроване програмне забезпечення для систем обробки електронних документів;
- досліджено та проаналізовано правила та методи відмінювання антропонімів українського та іноземного походження.

Практична значимість одержаних результатів полягає у тому, що розроблене програмне забезпечення може бути використане для систем обробки електронних документів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) BPE Business Processing for Enterprise [Електронний ресурс] / Maxim Sokhatsky // Toliman, 2015 – Режим доступу до ресурсу: https://n2o.dev/books/bpe.pdf;

2) N2O No Bullshit Sane Framework for Wild Web [Електронний ресурс] / Maxim Sokhatsky // Toliman, 2014 – Режим доступу до ресурсу: https://n2o.dev/books/n2o.pdf;

3) MAD Small and Fast Buil Tool for Erlang Apps [Електронний ресурс] – Режим доступу до ресурсу: https://n2o.dev/books/mad.pdf;

4) TWAIN Specification [Електронний ресурс] /the TWAIN Working Group // 2015 – Режим доступу до ресурсу: https://twain.org/wp-content/uploads/2017/03/TWAIN-2.4-Specification.pdf;

5) Kodak Twain [Електронний ресурс] – Режим доступу до ресурсу: https://pattersonsupport.custhelp.com/app/answers/detail/a_id/17942/~/kodak-twain;

6) Antroponimia ùemkowska na tùe poùskim i sùowackim XVI–XIX wiek / E. Wolnicz-Pawùowska // Warszawa, 1993;

7) З історії української антропонімії / Худаш М. Л.// 1977;

8) Український правопис / Українська національна комісія з питань правопису // Київ 2019;

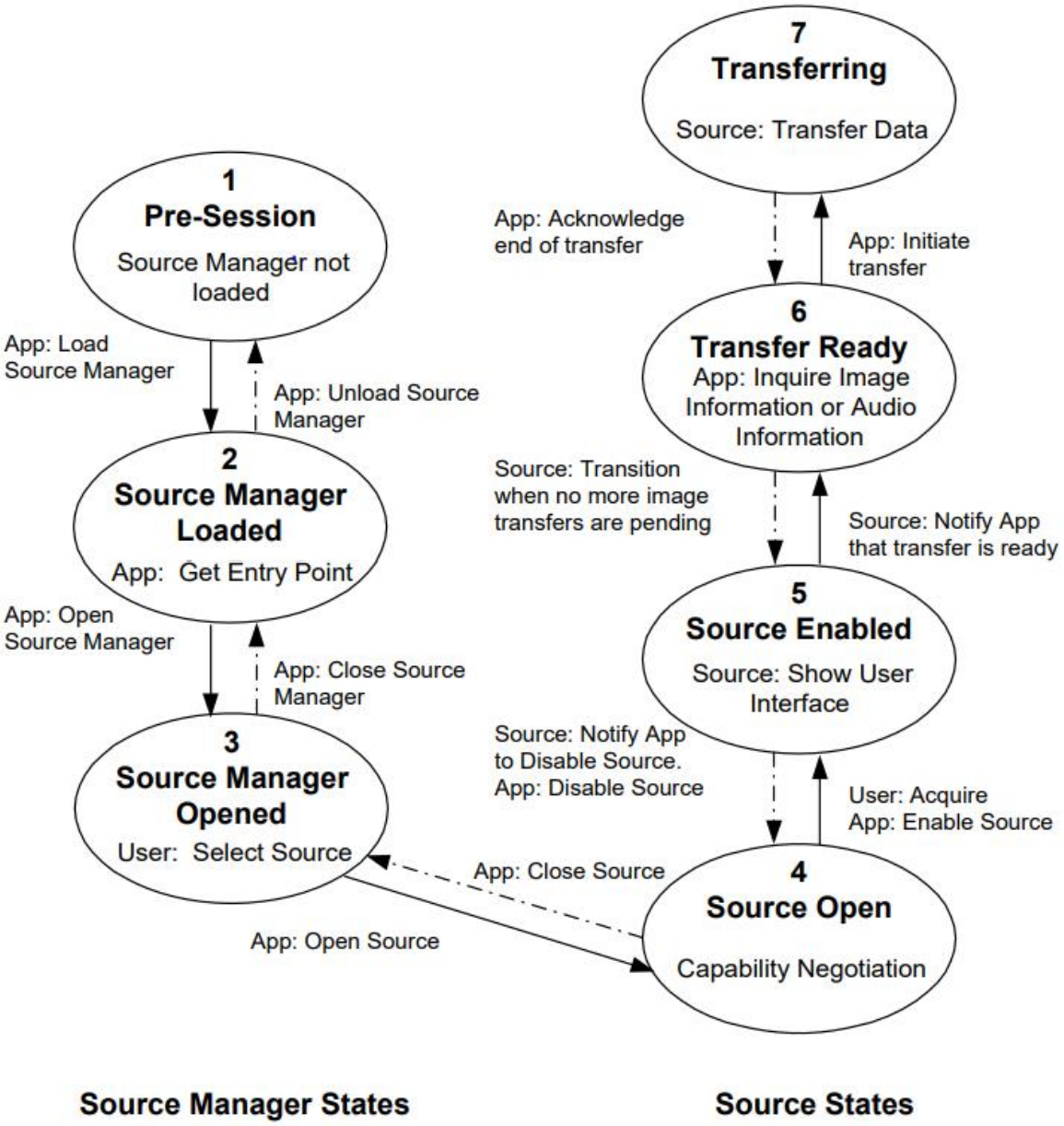
9) Erlang and OTP in Action / Matrin Logan // 2010;

10) Programing Erlang: Software for a Concurent Word / Joe Amstrong // 2013.

ДОДАТКИ

ДОДАТОК А

Діаграма переходу станів



ДОДАТОК Б

Лістинг коду

```
defmodule CRM.Templates.Engine do
  require ERP
  require NITRO
  require FORM
  require BPE
  alias CRM.Declension, as: Declension

  def generate(template, doc, opt) do
       case  :kvs.get('/crm/templates/#{:application.get_env(:crm,  :active_org,
"MIA:HSC")}/#{:erlang.element(1, doc)}', template) do
      {:ok, ERP.templateDoc(name: name, filePath: templateName, vars: fields)} -
>
        parameters = parameters(fields, doc, opt)
        cmd =  parameters |> cmd()
         guids = :lists.foldl(fn ERP.templateVar(tableGuid: []), acc -> acc ;
ERP.templateVar(tableGuid: guid), acc -> [guid|acc] end, [], parameters)
        {name, templateName, cmd, guids}
      _ -> {[], [], [], []}
    end
  end

  def parameters(fields, doc, opt) when is_tuple(doc) do
    :lists.map(fn
      ERP.templateVar(id: x, type: type, main: true, value: []) = var ->
        case :bpe.load(Keyword.get(opt, :main, [])) do
          BPE.process(docs: [mainDoc|_]) ->
            value = case :crm.pos(:erlang.element(1, mainDoc), x) do 0 -> ""; x
-> :erlang.element(x, mainDoc) end
            ERP.templateVar(var, tableGuid: type == :vector && :erp.guid || [],
value: value) |> normalizeValue
          _ -> ERP.templateVar(var, value: "\u200C")
        end
      ERP.templateVar(id: x, type: type, value: []) = var ->
         value = case :crm.pos(:erlang.element(1, doc), x) do 0 -> ""; x ->
:erlang.element(x, doc) end
       ERP.templateVar(var, tableGuid: type == :vector && :erp.guid || [], value:
value) |> normalizeValue
      any -> any
    end, fields)
  end

  def cmd(fields) do
    :lists.map(fn
```

```
      ERP.templateVar(id: id, type: :base64, value: value, field: field, options:
opt) ->
        text_cmd = case :proplists.get_value(:text, opt, []) do
          [] -> ""
                text  ->  value  in  [[],  ""]  &&  "  ,  base64  \"#{id}_text\"
\"#{:base64.encode("\u200C")}\""
            || " , base64 \"#{id}_text\" \"#{:base64.encode(text)}\""
        end
        value = is_tuple(value) && :kvs.field(value, field) || value
        value = value in [[], ""] && "\u200C" || value
       "base64 \"#{:nitro.to_binary(:form.atom([id, field] |> :lists.flatten))}\"
\"#{:base64.encode(value)}\"" <> text_cmd
        ERP.templateVar(id:  id,  type:  :vector,  value:  value,  field:  field,
tableName: tN, tableGuid: tG, options: opt) ->
        value = value |> :lists.flatten
        text_cmd = case :proplists.get_value(:text, opt, []) do
          [] -> ""
                text  ->  value  in  [[],  ""]  &&  "  ,  base64  \"#{id}_text\"
\"#{:base64.encode("\u200C")}\""
            || " , base64 \"#{id}_text\" \"#{:base64.encode(text)}\""
        end
           :file.write_file("priv/docx/#{tG}.csv",  (:lists.map(&:form.atom([id,
&1]), field) |> Enum.join(",")) <> "\n", [:append, :raw])
        :lists.foreach(&:file.write_file("priv/docx/#{tG}.csv", (:lists.map(fn f
-> :kvs.field(&1, f) end, field)
          |> Enum.join(",")) <> "\n", [:append, :raw]), value)
             value  in  [[],  ""]  &&  :file.write_file("priv/docx/#{tG}.csv",
(:lists.map(fn _ -> "\u200C" end, field)
          |> Enum.join(",")) <> "\n", [:append, :raw])
        "vector \"#{tN}\" \"priv/docx/#{tG}.csv\"" <> text_cmd
    end, fields)
  end

   def  normalizeValue(ERP.templateVar(type:  :base64,  value:  value,  callback:
callback) = nV) when callback != [], do:
      ERP.templateVar(nV,  value:  :lists.foldl(fn  {_,  func,  args},  v  ->
:erlang.apply(CRM.Templates.Engine, func, [v|args]) end, value, callback))
   def  normalizeValue(ERP.templateVar(type:  :vector,  value:  value,  callback:
callback) = nV) when callback != [], do:
    ERP.templateVar(nV, value: :lists.map(&:lists.foldl(fn {_, func, args}, v ->
:erlang.apply(CRM.Templates.Engine, func, [v|args]) end, &1, callback), value))
  def normalizeValue(nV), do: nV

def declension(ERP."Employee"(name: name, sex: sex) = e, d) do
    [l, f, _] = parse_cn(name)
    sex = sex |> :nitro.to_binary
       ERP."Employee"(e,  name:  "#{Declension.declension(l,  d,  sex,  true)}
#{Declension.declension(f, d, sex, false)}" |> :string.trim)
  end
```

```
  def declension(any), do: any

  def declensionPos(ERP."Employee"(position: pos) = e, d), do:
    ERP."Employee"(e, position: CRM.Declension.second_form_position(pos, d))
  def declensionPos(any), do: any
 def date_to_string({y, m, d}), do: :io_lib.format("~2..0w.~2..0w.~4..0w", [d, m,
y]) |> :lists.flatten() |> :nitro.to_binary()
  def date_to_string(_), do: ""

  def to_baze64(var, value) when is_binary(value), do:
    "base64 \"" <> :nitro.to_binary(var) <> "\" \"" <> :base64.encode(value) <>
"\""

  def to_baze64(var, _), do: "base64 \"" <> :nitro.to_binary(var) <> "\" \"" <>
:base64.encode("\u00A0") <> "\""
```

```
def    createTemplate(ERP.bizTask(),    pid),    do:    start({:generateTemplate,
:resolution, [], pid})
  def createTemplate(doc, pid), do:
    (case :kvs.field(doc, :template) do
      ERP.dict(id: i) -> start({:generateTemplate, i, [], pid});
      _ -> []
    end)

  def updateTemplate(_, _, changedFields, ERP.bizTask() = doc, pid) do
       case   :kvs.get('/crm/templates/#{:application.get_env(:crm,   :active_org,
"MIA:HSC")}/#{:erlang.element(1, doc)}', :resolution) do
      {:ok, ERP.templateDoc(id: i, vars: s)} ->
        :lists.any(fn ERP.templateVar(id: f) -> f in changedFields end, s) &&
        (:lists.foldl(fn
          ERP.fileDesc(template: []) = fd, {states, att} -> {states, [fd|att]}
           ERP.fileDesc(template: ERP.templateInfo(id: ^i, parent: x), hash: h),
{states, att} when h in [true, :finished] ->
            {[{:generateTemplate, i, x, pid}|states], att}
          ERP.fileDesc(id: x, template: ERP.templateInfo(id: ^i)), {states, att}
->
            {[{:generateTemplate, i, x, pid}|states], att}
           ERP.fileDesc(template: ERP.templateInfo(id: t)), {states, att} when t
!= i ->
            {[{:generateTemplate, i, t, pid}|states], att}
          fd, {states, att} -> {states, [fd|att]}
         end, {[], []}, :kvs.field(doc, :attachments))) || {[], :kvs.field(doc,
:attachments)}
      _ -> {[], :kvs.field(doc, :attachments)}
    end
  end
  def updateTemplate(ERP.templateDoc(id: i1), ERP.templateDoc(id: i2), _, doc,
pid) when i1 != i2, do:
```

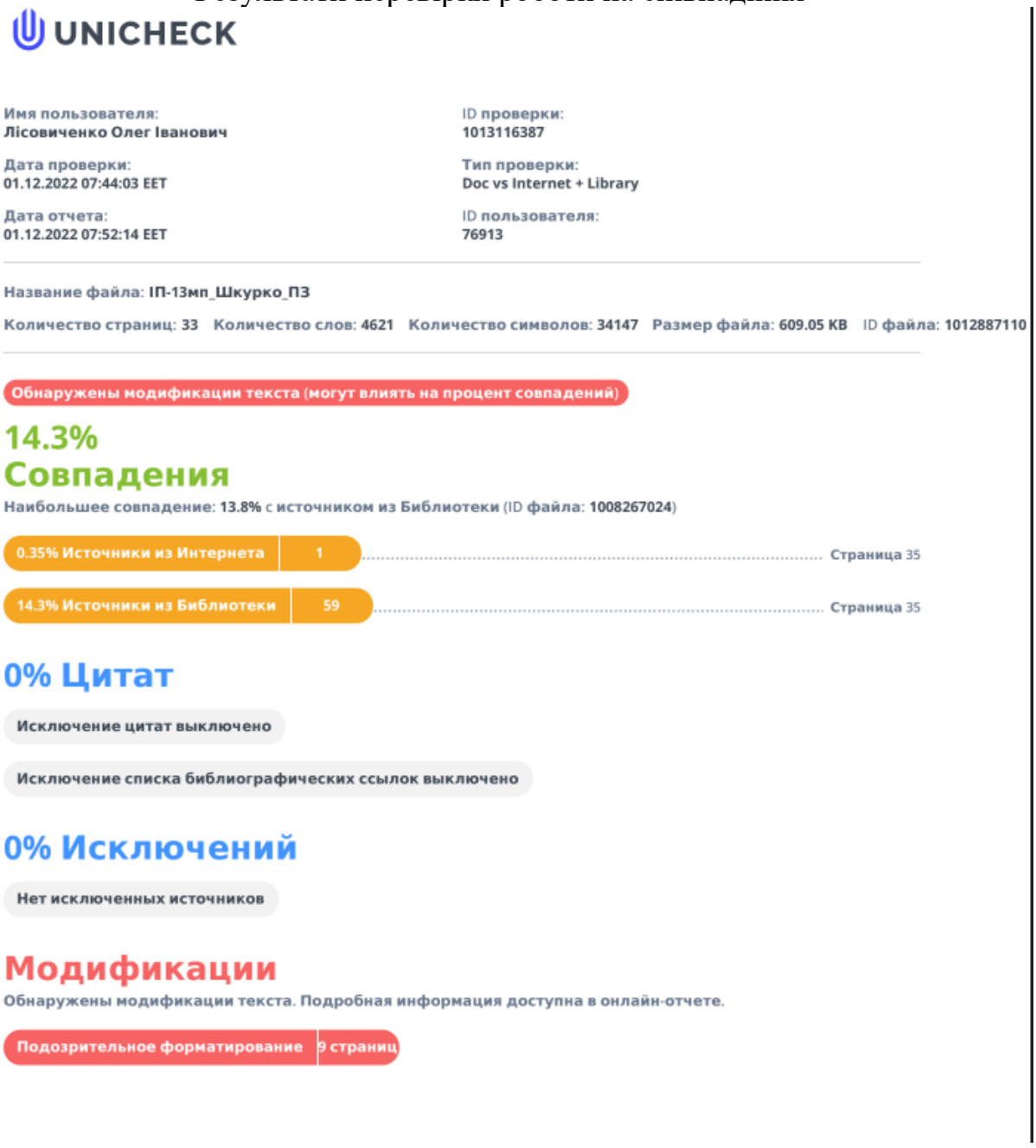
```
    {[{:generateTemplate, i1, [], pid}], :lists.filter(fn ERP.fileDesc(template:
ERP.templateInfo(id:  t))  ->  t  !=  i2;  _  ->  true  end,  :kvs.field(doc,
:attachments))}
   def  updateTemplate(ERP.templateDoc(id:  i,  vars:  s),  ERP.templateDoc(),
changedFields, doc, pid) do
    :lists.any(fn ERP.templateVar(id: f) -> f in changedFields end, s) &&
    (:lists.foldl(fn
       ERP.fileDesc(id: guid, template: ERP.templateInfo(id: ^i)), {states, att}
-> {[{:generateTemplate, i, guid, pid}|states], att}
      fd, {states, att} ->{states, [fd|att]}
     end, {[], []}, :kvs.field(doc, :attachments))) || {[], :kvs.field(doc,
:attachments)}
  end
  def updateTemplate(_, ERP.templateDoc(id: i), _, doc, _), do:
    {[], :lists.filter(fn ERP.fileDesc(template: ERP.templateInfo(id: t)) -> t !=
i; _ -> true end, :kvs.field(doc, :attachments))}
  def updateTemplate(ERP.templateDoc(id: i), _, _, doc, pid), do:
    {[{:generateTemplate, i, [], pid}], :kvs.field(doc, :attachments)}
  def updateTemplate(_, _, _, doc, _), do: {[], :kvs.field(doc, :attachments)}
def start(request) do
    pi = N2O.pi(module: TemplatePI, restart: :transient, table: :async, sup: :n2o,
state: request, timeout: 60000, name: "template_pi")
    case :n2o_pi.start(pi) do
      {:error, {:already_started, x}} -> send(x, request)
      {x, _} when is_pid(x) -> send(x, request)
      x -> x
    end
  end

  def proc(:init, N2O.pi() = pi), do: {:ok, N2O.pi(pi, state: [])}

  def proc({:generateTemplate, template, prevGuid, pid}, pi) do
    guid = :erp.guid()
    BPE.process(docs: [doc | t], options: opt) = proc = :bpe.load(pid)
                {name,       templateName,       data,       csv_guids}       =
CRM.Templates.Engine.generate(template, doc, opt)
    generateTemplate(templateName, "templates", guid, prevGuid, data)
    :lists.foreach(&File.rm("priv/docx/#{&1}.csv"), csv_guids)

    size =
      case :file.read_file_info("priv/templates/" <> guid) do
        {:ok, info} -> File.Stat.from_record(info).size
        {:error, _} -> 0
      end
    mime = "docx"
    localTime = :calendar.local_time()
    user = :n2o.user()

    fd =
```

```
      ERP.fileDesc(
        id: guid,
        fileName: name,
        parent_id: prevGuid,
        desc: [],
        type: :template,
        mime: mime,
        size: size,
        loaded: localTime,
        lastModified: localTime,
        creator: user,
        lastModifier: user,
        template: ERP.templateInfo(id: template, parent: guid),
        needSign: true
      )

    root = CRM.FileUpload.root()

    pathBefore = Path.join([File.cwd!(), "priv", "templates", guid <> ".docx"])
    pathAfter = Path.join(root, guid)

    :ok = File.mkdir_p(root)

    case File.copy(pathBefore, pathAfter) do
      {:ok, _} ->
        old_a = :kvs.field(doc, :attachments)
        main = :lists.keyfind(true, :crm.pos(:fileDesc, :main), old_a) == false
         newDoc = :kvs.setfield(doc, :attachments, [ERP.fileDesc(fd, main: main)
| old_a])
        newProc = BPE.process(proc, docs: [newDoc | t])

        :bpe.persist(pid, newProc, [
                  BPE.continue(fn:   :updateMonitor,   type:   :spawn,   module:
:"Elixir.General.Proc", args: [[], newProc, []])
        ])

        File.rm("priv/templates/" <> to_string(guid) <> ".docx")
        File.rm("priv/docx/in" <> to_string(guid) <> ".docx")

        CRM.info 'DOC Complete Template ~p', [{guid, pid}]

      {:error, code} ->
        path = Path.join(["priv", "templates", guid <> ".docx"])
        CRM.info 'DOC Complete Template ~p', [{posix(path, code), guid, pid}]
    end

    {:noreply, pi}
  end
defp generateTemplate(templateName, dir, guid, prevGuid, data), do:
```

```
    (template = case prevGuid do [] -> "priv/#{dir}/#{templateName}.docx"; x ->
"priv/storage/#{x}" end;
     err = CRM.QR.execute('cp "#{template}" "priv/docx/in#{guid}.docx"', false);
       CRM.QR.execute('priv/docx/docx  -vars  #{Enum.join(data,  "  ,  ")}  -in
"priv/docx/in#{guid}.docx"' ++
                    '-out "priv/#{dir}/#{guid}.docx"', err))
def updateTemplate(oldGuid, newGuid, data), do:
               (err      =      CRM.QR.execute('cp      "priv/storage/'      ++
:unicode.characters_to_list(oldGuid) ++ '" "priv/docx/'
                          ++ :unicode.characters_to_list(newGuid) ++ '.docx" ',
false);
    CRM.QR.execute('priv/docx/docx -vars ' ++ :unicode.characters_to_list(data)
++ ' -in "priv/docx/' ++ :unicode.characters_to_list(newGuid)
                                      ++   '.docx"   -out   "priv/storage/'   ++
:unicode.characters_to_list(newGuid) ++ '.docx"', err))
```

ДОДАТОК В

Результати перевірки роботи на співпадіння



UNICHECK

Имя пользователя:
Лісовиченко Олег Іванович

ID проверки:
1013116387

Дата проверки:
01.12.2022 07:44:03 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
01.12.2022 07:52:14 EET

ID пользователя:
76913

Название файла: ІП-13мп_Шкурко_ПЗ

Количество страниц: 33 Количество слов: 4621 Количество символов: 34147 Размер файла: 609.05 KB ID файла: 1012887110

Обнаружены модификации текста (могут влиять на процент совпадений)

14.3%
Совпадения

Наибольшее совпадение: 13.8% с источником из Библиотеки (ID файла: 1008267024)

0.35% Источники из Интернета 1 ..... Страница 35

14.3% Источники из Библиотеки 59 ..... Страница 35

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников

Модификации

Обнаружены модификации текста. Подробная информация доступна в онлайн-отчете.

Подозрительное форматирование 9 страниц